LLNL-TR-462325

Adaptive and Efficient Computing for Subsurface Simulation within ParFlow

H. Tiedeman, C. S. Woodward

November 16, 2010

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Adaptive and Efficient Computing for Subsurface Simulation within ParFlow LDRD-10-ERD-011[*]

Hilari Tiedeman[†] Carol S. Woodward[‡]

November 15, 2010

1 Introduction

This project is concerned with the PF.WRF model as a means to enable more accurate predictions of wind fluctuations and subsurface storage. As developed at LLNL, PF.WRF couples a groundwater (subsurface) and surface water flow model (ParFlow) to a mesoscale atmospheric model (WRF, Weather Research and Forecasting Model). It was developed as a unique tool to address coupled water balance and wind energy questions that occur across traditionally separated research regimes of the atmosphere, land surface, and subsurface. PF.WRF is capable of simulating fluid, mass, and energy transport processes in groundwater, vadose zone, root zone, and land surface systems, including overland flow [4, 5], and allows for the WRF model to both directly drive and respond to surface and subsurface hydrologic processes and conditions.

The current PF.WRF model is constrained to have uniform spatial gridding below the land surface and matching areal grids with the WRF model at the land surface. There are often cases where it is advantageous for land surface, overland flow and subsurface models to have finer gridding than their atmospheric counterparts. Finer vertical discretization is also advantageous near the land surface (to properly capture feedbacks) yet many applications have a large vertical extent. However, the surface flow is strongly dependent on topography leading to a need for greater lateral resolution in some regions and the subsurface flow is tightly coupled to the atmospheric model near the surface leading to a need for finer vertical resolution. In addition, the interactions (e.g. rain) will be highly variable in space and time across the problem domain so an adaptive scheme is preferred to a static strategy to efficiently use computing and memory resources. As a result, this project focussed on algorithmic research required for development of an adaptive simulation capability in the PF.WRF system and its subsequent use in an application problem in the Central Valley of California.

This report documents schemes of use for a future implementation of an adaptive grid capability within the ParFlow subsurface flow simulator in PF.WRF. The methods describe specific handling of the coarse/fine boundaries within a cell-centered discretization of the nonlinear

[*]This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work was funded by the Laboratory Directed Research and Development Program at LLNL under project tracking code LDRD-10-ERD-011.

[†]Department of Mathematics, Southern Methodist University, Dallas, TX (`htiedeman@smu.edu`)

[‡]Lawrence Livermore National Laboratory, Center for Applied Scientific Computing, PO Box 808, L-561, Livermore, CA 94551, (`cswoodward@llnl.gov`)

parabolic Richards' equation model for variable saturated flow. In addition, we describe development of a spline fit and table lookup method implemented within ParFlow to enhance computational efficiency of variably saturated flow calculations.

The rest of this document is organized as follows. Section 2 details the findings of appropriate AMR methods for use with Richards' equation, and section 3 discusses the work done to implement a spline fit to data for the relative permeability curves. Section **??** discusses the specification of the Central Valley site, and the last section gives some concluding remarks about the current state of the ParFlow code.

2 Adaptive Mesh Refinement Discretization for Richards' Equation

In ParFlow, discretization of the Richards' equation is done with a uniform mesh across each dimension of space. The mesh size must be small to capture the effects of groundwater flow near riverbeds. However, this causes a fine mesh size for the entire computational domain. Our goal is to discretize Richards' Equation to use adaptive mesh refinement (AMR), allowing for a fine mesh where it is needed (near the surface) and a coarser mesh where it can save calculations and computer memory (away from the surface).

2.1 Discretization of Richards' Equation

ParFlow employs the Richards' equation model for variably saturated flow. This model is written as, [6]:

$$S(p)S_s\frac{\partial p}{\partial t} + \frac{\partial(S(p)\rho(p)\phi)}{\partial t} - \nabla\cdot(\mathbf{K}(p)\rho(p)(\nabla p - \rho(p)\vec{g})) = Q, \tag{1}$$

in Ω, where S is the water saturation, p is the pressure head of water, S_s is the specific storage coefficient, ρ is the density, ϕ is the porosity of the medium, $\mathbf{K}(p)$ is the hydraulic conductivity tensor, $\vec{g}$ is the gravity vector, and Q is the water source/sink term. The hydraulic conductivity can be broken down into:

$$\mathbf{K}(p) = \frac{\bar{k}k_r(p)}{\mu},$$

where $\bar{k}$ is the intrinsic permeability, k_r is the relative permeability, and μ is the viscosity. The relative permeability is a function of the pressure head and is defined by the Van Genuchten function:

$$k_r(p) = \frac{\left(1 - \frac{(\alpha p)^{n-1}}{(1+(\alpha p)^n)^m}\right)^2}{(1+(\alpha p)^n)^{m/2}}.$$

The boundary conditions are:

$$\begin{aligned} p &= p_D, \text{ on } \Gamma^D \\ -\mathbf{K}(p)\nabla p\cdot\mathbf{n} &= g_N, \text{ on } \Gamma^N, \end{aligned}$$

where $\Gamma^D \cup \Gamma^N = \partial\Omega,\ \Gamma^D \neq \emptyset$, and $\vec{n}$ is an outward pointing, unit, normal vector to Ω. The initial condition is given by:

$$p = p^0(x),\ t = 0.$$

Let LHS denote the left-hand side of Richards' equation:

$$LHS = S(p)S_s\frac{\partial p}{\partial t} + \frac{\partial(S(p)\rho(p)\phi)}{\partial t}, \tag{2}$$

and let RHS denote the right-hand side:

$$RHS = \nabla\cdot(\mathbf{K}(p)\rho(p)(\nabla p - \rho(p)\vec{g})) + Q. \tag{3}$$

We can discretize (2) using an implicit backward Euler method in time:

$$LHS = S(p^n_{i,j,k})S_{s_{i,j,k}}\frac{p^{n+1}_{i,j,k} - p^n_{i,j,k}}{\Delta t} + \phi_{i,j,k}\frac{S(p^{n+1}_{i,j,k})\rho(p^{n+1}_{i,j,k}) - S(p^n_{i,j,k})\rho(p^n_{i,j,k})}{\Delta t}, \tag{4}$$

We can discretize (3) by using an implicit cell-centered finite difference method in space:

$$\begin{aligned} RHS &= \left[\bar{k}(x)_{i+\frac{1}{2},j,k}k_r(p^n_{i+\frac{1}{2},j,k})\frac{p^{n+1}_{i+1,j,k} - p^{n+1}_{i,j,k}}{\Delta x} - \bar{k}(x)_{i-\frac{1}{2},j,k}k_r(p^n_{i-\frac{1}{2},j,k})\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i-1,j,k}}{\Delta x}\right]\frac{1}{\Delta x} \\ &+ \left[\bar{k}(x)_{i,j+\frac{1}{2},k}k_r(p^{n+1}_{i,j+\frac{1}{2},k})\frac{p^{n+1}_{i,j+1,k} - p^{n+1}_{i,j,k}}{\Delta y} - \bar{k}(x)_{i,j-\frac{1}{2},k}k_r(p^n_{i,j-\frac{1}{2},k})\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i,j-1,k}}{\Delta y}\right]\frac{1}{\Delta y} \\ &+ \left[\bar{k}(x)_{i,j,k+\frac{1}{2}}k_r(p^n_{i,j,k+\frac{1}{2}})\left(\frac{p^{n+1}_{i,j,k+1} - p^{n+1}_{i,j,k}}{\Delta z} - \rho_z(p^{n+1}_{i,j,k+\frac{1}{2}})g\right)\right. \\ &- \left.\bar{k}(x)_{i,j,k-\frac{1}{2}}k_r(p^n_{i,j,k-\frac{1}{2}})\left(\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i,j,k-1}}{\Delta z} - \rho_z(p^{n+1}_{i,j,k-\frac{1}{2}})g\right)\right]\frac{1}{\Delta z} + Q^{n+1}_{i,j,k}. \end{aligned} \tag{5}$$

Harmonic averaging and upwinding will be used to determine the interface coefficients of the intrinsic permeability, $\bar{k}(x)$, and the relative permeability, $k_r(p)$, respectively. Combining (4) and (5), we get:

$$\begin{aligned} F(p^{n+1}_{i,j,k}) &= \Delta x\Delta y\Delta z\left[S(p^n_{i,j,k})S_{s_{i,j,k}}(p^{n+1}_{i,j,k} - p^n_{i,j,k}) + \phi_{i,j,k}(S(p^{n+1}_{i,j,k})\rho(p^{n+1}_{i,j,k}) - S(p^n_{i,j,k})\rho(p^n_{i,j,k}))\right] \\ &- \Delta t\Delta y\Delta z\left[\bar{k}(x)_{i+\frac{1}{2},j,k}k_r(p^n_{i+\frac{1}{2},j,k})\frac{p^{n+1}_{i+1,j,k} - p^{n+1}_{i,j,k}}{\Delta x} - \bar{k}(x)_{i-\frac{1}{2},j,k}k_r(p^n_{i-\frac{1}{2},j,k})\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i-1,j,k}}{\Delta x}\right] \\ &- \Delta t\Delta x\Delta z\left[\bar{k}(x)_{i,j+\frac{1}{2},k}k_r(p^{n+1}_{i,j+\frac{1}{2},k})\frac{p^{n+1}_{i,j+1,k} - p^{n+1}_{i,j,k}}{\Delta y} - \bar{k}(x)_{i,j-\frac{1}{2},k}k_r(p^n_{i,j-\frac{1}{2},k})\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i,j-1,k}}{\Delta y}\right] \\ &- \Delta t\Delta x\Delta y\left[\bar{k}(x)_{i,j,k+\frac{1}{2}}k_r(p^n_{i,j,k+\frac{1}{2}})\left(\frac{p^{n+1}_{i,j,k+1} - p^{n+1}_{i,j,k}}{\Delta z} - \rho_z(p^{n+1}_{i,j,k+\frac{1}{2}})g\right)\right. \\ &- \left.\bar{k}(x)_{i,j,k-\frac{1}{2}}k_r(p^n_{i,j,k-\frac{1}{2}})\left(\frac{p^{n+1}_{i,j,k} - p^{n+1}_{i,j,k-1}}{\Delta z} - \rho_z(p^{n+1}_{i,j,k-\frac{1}{2}})g\right)\right] - \Delta t\Delta x\Delta y\Delta z Q^{n+1}_{i,j,k} = 0 \end{aligned} \tag{6}$$

2.2 Richards' Equation with AMR

The discretization in (6) will present a problem when a fine cell shares an edge with a coarse cell. If (i, j, k) is adjacent to a coarse cell, then the coarse cell's center is not aligned with (i, j, k)'s center. If the coarse cell data was to be used as in teh scheme above, an order of accuracy would be lost. This paper will introduce two interpolation schemes (linear and quadratic [7]) to create a fine ghost node that uses both coarse and fine data to approximate what the data will look like in the ghost cell. To simplify this paper, examples of these calculations will be shown in 2D on only a portion of (6). The following discretization will be used:

$$\begin{aligned}(\nabla \cdot (K(x)\nabla P))_{if,jf} &= \frac{K_{if+\frac{1}{2},jf}(P_{if+1,jf} - P_{if,jf})}{{h_x}^2} - \frac{K_{if-\frac{1}{2},jf}(P_{if,jf} - P_{if-1,jf})}{{h_x}^2} \\ &+ \frac{K_{if,jf+\frac{1}{2}}(P_{if,jf+1} - P_{if,jf})}{{h_y}^2} - \frac{K_{if,jf-\frac{1}{2}}(P_{if,jf} - P_{if,jf-1})}{{h_y}^2}. \qquad (7)\end{aligned}$$

We will show how to modify this function for point (if, jf) at coarse/fine interfaces based on Figure 1.

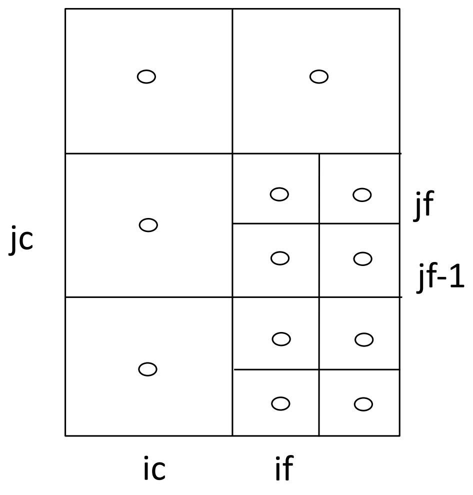

Figure 1: AMR grid with two levels.

2.2.1 Linear Interpolation

First, linear interpolation will be used in order to calculate the x derivative of the point $(if - \frac{1}{2}, jf)$ (the boundary between (ic, jc) and (if, jf), aligned with (if, jf), see Figure 2). Taylor series expansions will need to be calculated for $u_{if,jf}$, $u_{ic,jc}$, and $u_{if+1,jf}$ in terms of $u_{if-\frac{1}{2},jf}$:

$$u_{if,jf} = u_{if-\frac{1}{2},jf} + \frac{1}{2}h_x\partial_x u_{if-\frac{1}{2},jf+} + \frac{1}{8}h_x^2\partial_x^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({h_x}^3), \qquad (8)$$

$$\begin{aligned}u_{ic,jc} &= u_{if-\frac{1}{2},jf} - \frac{1}{2}H_x\partial_x u_{if-\frac{1}{2},jf} - \frac{1}{4}H_y\partial_y u_{if-\frac{1}{2},jf} + \frac{1}{8}H_x^2\partial_x^2 u_{if-\frac{1}{2},jf} \\ &+ \frac{1}{8}H_xH_y\partial_x\partial_y u_{if-\frac{1}{2},jf} + \frac{1}{32}H_y^2\partial_y^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({H_x}^3) + \mathcal{O}({H_y}^3), \qquad (9)\end{aligned}$$

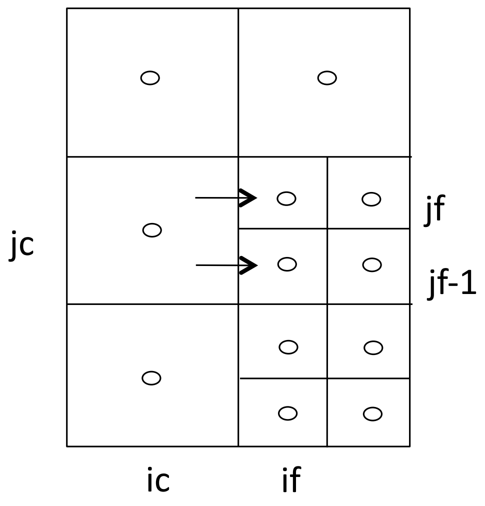

Figure 2: Linear interpolation consists of creating and combining Taylor series of the data at the coarse cell and two fine cells in terms of the boundary between them to determine the flux at the interface.

$$\begin{aligned} u_{if,jf-1} &= u_{if-\frac{1}{2},jf} + \frac{1}{2}h_x\partial_x u_{if-\frac{1}{2},jf} - h_y\partial_y u_{if-\frac{1}{2},jf} + \frac{1}{8}h_x^2\partial_x^2 u_{if-\frac{1}{2},jf} \\ &- \frac{1}{2}h_xh_y\partial_x\partial_y u_{if-\frac{1}{2},jf} + \frac{1}{2}h_y^2\partial_y^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({h_x}^3) + \mathcal{O}({h_y}^3). \end{aligned} \tag{10}$$

Subtracting (9) from (8) gives

$$\begin{aligned} u_{if,jf} - u_{ic,jc} &= \frac{H_x + h_x}{2}\partial_x u_{if-\frac{1}{2},jf} + \frac{1}{4}H_y\partial_y u_{if-\frac{1}{2},jf} + \frac{h_x^2 - H_x^2}{8}H_x^2\partial_x^2 u_{if-\frac{1}{2},jf} \\ &- \frac{1}{8}H_xH_y\partial_x\partial_y u_{if-\frac{1}{2},jf} - \frac{1}{32}H_y^2\partial_y^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({H_x}^3) + \mathcal{O}({H_y}^3) + \mathcal{O}({h_x}^3). \end{aligned}$$

Appropriate calculations will give an approximation of the first derivative of $u_{if-\frac{1}{2},jf}$ with respect to x:

$$\begin{aligned} \partial_x u_{if-\frac{1}{2},jf} &= \frac{2(u_{if,jf} - u_{ic,jc})}{H_x + h_x} - \frac{H_y}{2(H_x + h_x)}\partial_y u_{if-\frac{1}{2},jf} \\ &+ \mathcal{O}(H_x - h_x) + \mathcal{O}\left(\frac{H_xH_y}{H_x + h_x}\right) + \mathcal{O}\left(\frac{H_y^2}{H_x + h_x}\right). \end{aligned} \tag{11}$$

Now an approximation of the derivative of $u_{if-\frac{1}{2},jf}$ with respect to y is needed to finish the scheme. To begin, subtract (10) from (8) to yield

$$\begin{aligned} u_{if,jf} - u_{if,jf-1} &= h_y\partial_y u_{if-\frac{1}{2},jf} + \frac{1}{2}h_xh_y\partial_x\partial_y u_{if-\frac{1}{2},jf} \\ &- \frac{1}{2}h_y^2\partial_y^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({h_y}^3). \end{aligned} \tag{12}$$

Appropriate calculations will give an approximation of the first derivative of $u_{if-\frac{1}{2},jf}$ with respect to y,

$$\partial_y u_{if-\frac{1}{2},jf} = \frac{u_{if,jf} - u_{if,jf-1}}{h_y} + \mathcal{O}(h_x) + \mathcal{O}(h_y). \tag{13}$$

Substituting (13) and (11) will give an approximation of the first derivative of $u_{if-\frac{1}{2},jf}$ with respect to x,

$$\begin{aligned}\partial_x u_{if-\frac{1}{2},jf} &= \frac{2(u_{if,jf}-u_{ic,jc})}{H_x+h_x} - \frac{H_y(u_{if,jf}-u_{if,jf-1})}{2h_y(H_x+h_x)} \\ &+ \mathcal{O}(H_x-h_x) + \mathcal{O}\left(\frac{H_xH_y}{H_x+h_x}\right) + \mathcal{O}\left(\frac{H_y^2}{H_x+h_x}\right). \end{aligned} \tag{14}$$

Now, the y derivative of $u_{if,jf+\frac{1}{2}}$ will be calculated using linear interpolation. To do this, Taylor series expansions will need to be computed for $u_{if,jf}$, $u_{ic+1,jc+1}$, and $u_{if+1,jf}$ in terms of $u_{if,if+\frac{1}{2}}$ (the boundary between $(ic+1, jc+1)$ and (if, jf), aligned with (if, jf)). This gives,

$$u_{if,jf} = u_{if,jf+\frac{1}{2}} - \frac{1}{2}h_y\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{8}h_y^2\partial_y^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({h_y}^3), \tag{15}$$

$$\begin{aligned} u_{ic+1,jc+1} &= u_{if,jf+\frac{1}{2}} + \frac{1}{2}H_y\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{4}H_x\partial_x u_{if,jf+\frac{1}{2}} + \frac{1}{8}H_y^2\partial_y^2 u_{if,jf+\frac{1}{2}} \\ &+ \frac{1}{8}H_xH_y\partial_x\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{32}H_x^2\partial_x^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({H_y}^3) + \mathcal{O}({H_x}^3), \end{aligned} \tag{16}$$

and

$$\begin{aligned} u_{if+1,jf} &= u_{if,jf+\frac{1}{2}} - \frac{1}{2}h_y\partial_y u_{if,jf+\frac{1}{2}} + h_x\partial_x u_{if,jf+\frac{1}{2}} + \frac{1}{8}h_y^2\partial_y^2 u_{if,jf+\frac{1}{2}} \\ &- \frac{1}{2}h_xh_y\partial_x\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{2}h_x^2\partial_x^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({h_y}^3) + \mathcal{O}({h_x}^3). \end{aligned} \tag{17}$$

Subtracting (16) from (15) yields,

$$\begin{aligned} u_{if,jf} - u_{ic+1,jc+1} &= -\frac{H_y+h_y}{2}\partial_y u_{if,jf+\frac{1}{2}} - \frac{1}{4}H_x\partial_x u_{if,jf+\frac{1}{2}} + \frac{h_y^2-H_y^2}{8}\partial_y^2 u_{if,jf+\frac{1}{2}} \\ &- \frac{1}{8}H_xH_y\partial_x\partial_y u_{if,jf+\frac{1}{2}} - \frac{1}{32}H_x^2\partial_x^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({H_y}^3) + \mathcal{O}({H_x}^3) + \mathcal{O}({h_y}^3). \end{aligned}$$

Appropriate calculations will yield an approximation of the first derivative of $u_{if,jf+\frac{1}{2}}$ with respect to y,

$$\begin{aligned} \partial_y u_{if,jf+\frac{1}{2}} &= -\frac{2(u_{if,jf}-u_{ic+1,jc+1})}{H_y+h_y} - \frac{H_x}{2(H_y+h_y)}\partial_x u_{if,jf+\frac{1}{2}} \\ &+ \mathcal{O}(H_y-h_y) + \mathcal{O}\left(\frac{H_xH_y}{H_y+h_y}\right) + \mathcal{O}\left(\frac{H_x^2}{H_y+h_y}\right). \end{aligned} \tag{18}$$

An approximation of the first derivative of $u_{if,jf+\frac{1}{2}}$ with respect to x is needed to complete the scheme. Subtracting (17) from (15) will give,

$$\begin{aligned} u_{if,jf} - u_{if+1,jf} &= -h_x\partial_x u_{if,jf+\frac{1}{2}} + \frac{1}{2}h_xh_y\partial_x\partial_y u_{if,jf+\frac{1}{2}} \\ &- \frac{1}{2}h_x^2\partial_x^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({h_x}^3). \end{aligned}$$

Appropriate calculations will give an approximation of the first derivative of $u_{if,jf+\frac{1}{2}}$ with respect to x,

$$\partial_x u_{if,jf+\frac{1}{2}} = -\frac{u_{if,jf}-u_{if+1,jf}}{h_x} + \mathcal{O}(h_x) + \mathcal{O}(h_y). \tag{19}$$

Combining (18) and (19) gives the first order approximation of the first derivative of $u_{if,jf+\frac{1}{2}}$ with respect to y,

$$\begin{aligned}\partial_y u_{if,jf+\frac{1}{2}} &= -\frac{2(u_{if,jj} - u_{ic+1,jc+1})}{H_y + h_y} + \frac{H_x(u_{if,jf} - u_{if+1,jf})}{2h_x(H_y + h_y)} \partial_x u_{if,jf+\frac{1}{2}} \\ &+ \mathcal{O}(H_y - h_y) + \mathcal{O}\left(\frac{H_x H_y}{H_y + h_y}\right) + \mathcal{O}\left(\frac{H_x^2}{H_y + h_y}\right). \qquad (20)\end{aligned}$$

The approximations of the derivatives in (14) and (20) can be substituted into the discretization scheme of equation (7). The approximation of $\partial_x u_{if-\frac{1}{2},jf}$ found in (14) will replace:

$$\frac{P_{if,jf} - P_{if-1,jf}}{h_x},$$

while the approximation of $\partial_y u_{if,jf+\frac{1}{2}}$ found in (20) will replace

$$\frac{P_{if,jf+1} - P_{if,jf}}{h_y}.$$

The final discretization scheme using linear interpolation at the coarse/fine interface will be:

$$\begin{aligned} FD_{if,jf} &= \frac{K_{if+\frac{1}{2},jf}(P_{if+1,jf} - P_{if,jf})}{{h_x}^2} \\ &- \frac{K_{if-\frac{1}{2},jf}}{h_x} \frac{(4h_y - H_y)u_{if,jf} + H_y u_{if,jf-1} - 4h_y u_{ic,jc}}{2h_y(H_x + h_y)} \\ &+ \frac{K_{if,jf+\frac{1}{2}}}{h_y} \frac{-(4h_x - H_x)u_{if,jf} - H_x u_{if+1,jf} + 4h_x u_{ic+1,jc+1}}{2h_x(H_y + h_y)} \\ &- \frac{K_{if,jf-\frac{1}{2}}(P_{if,jf} - P_{if,jf-1})}{{h_y}^2}, \qquad (21)\end{aligned}$$

where

$$\begin{aligned}(\nabla \cdot (K(x)\nabla P))_{if,jf} &= FD_{if,jf} + \mathcal{O}\left(\frac{H_x - h_x}{h_x}\right) + \mathcal{O}\left(\frac{H_y - h_y}{h_y}\right) \\ &+ \mathcal{O}\left(\frac{H_x H_y}{h_x(H_x + h_x)}\right) + \mathcal{O}\left(\frac{H_x H_y}{h_y(H_y + h_y)}\right) \\ &+ \mathcal{O}\left(\frac{H_y^2}{h_x(H_x + h_x)}\right) + \mathcal{O}\left(\frac{H_x^2}{h_y(H_y + h_y)}\right).\end{aligned}$$

2.2.2 Quadratic Interpolation

For quadratic interpolation, a two stage process is done. The first stage will fit a quadratic interpolant to the three coarse cells that share the appropriate edge and corners with the interface that is being approximated. The second stage will fit another quadratic interpolant through the two fine grid points that are normal to the original interpolant and the point on the original interpolant aligned with the two fine nodes (see Figure 3). An x derivative

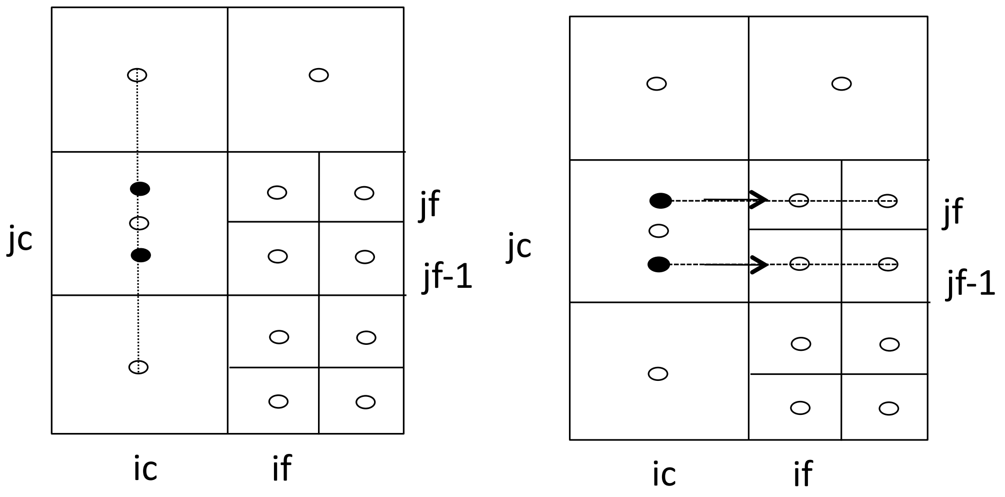

Figure 3: Process of quadratic interpolation. Left: Create a quadratic interpolant through the three coarse nodes to get a value that aligns with the fine cell data. Right: Use the interpolated value and the fine cell data to find data at the fine ghost node.

approximation of the interface node $(if - \frac{1}{2}, jf)$ will be obtained first. A quadratic interpolant of the three coarse nodes is given by

$$\begin{aligned} L(\tau) &= \sum_{j=0}^{2} \left(u_j \cdot \prod_{f=0, f\neq j}^{2} \frac{\tau - \tau_f}{\tau_j - \tau_f} \right) \\ &= u_{ic,jc+1} \frac{\tau(\tau + 2H_y)}{2{H_y}^2} - u_{ic,jc} \frac{(\tau - H_y)(\tau + H_y)}{{H_y}^2} + u_{ic,jc-1} \frac{\tau(\tau - H_y)}{2{H_y}^2}, \end{aligned}$$

where $\tau = 0$ is located at the center data point, $u_{ic,jc}$. The point in this interpolant aligned with the fine grid data, $(ic, jc + \frac{1}{4})$, will be obtained by evaluating $L(\tau)$ at $\tau = \frac{H_y}{4}$,

$$u_{ic,jc+\frac{1}{4}} = L\left(\frac{H_y}{4}\right) = \frac{5}{32} u_{ic,jc+1} + \frac{15}{16} u_{ic,jc} - \frac{3}{32} u_{ic,jc-1}. \tag{22}$$

Data from this point, $u_{if,jf}$ and $u_{if+1,jf}$ will be used to produce a quadratic interpolant through the fine data,

$$\begin{aligned} L(\eta) &= \sum_{j=0}^{2} \left(u_j \cdot \prod_{f=0, f\neq j}^{2} \frac{\eta - \eta_f}{\eta_j - \eta_f} \right) \\ &= u_{ic,jc+\frac{1}{4}} \frac{(2\eta - h_x)(2\eta - 3h_x)}{(H_x + h_x)(H_x + 3h_x)} - u_{if,jf} \frac{(2\eta + H_x)(2\eta - 3h_x)}{2h_x(H_x + h_x)} \\ &+ u_{if+1,jf} \frac{(2\eta + H_x)(2\eta - h_x)}{2h_x(H_x + 3h_x)}, \end{aligned}$$

where $\eta = 0$ is located at the coarse/fine interface, $(if - \frac{1}{2}, jf)$. Taking the derivative of this function and evaluating at $\eta = 0$ will give a first derivative approximation of the interface,

$$\begin{aligned}
\partial_x u_{if-\frac{1}{2},jf} = L'(0) &= -\frac{8h_x}{(H_x+h_x)(H_x+3h_x)} u_{ic,jc+\frac{1}{4}} \\
&- \frac{H_x - 3h_x}{h_x(H_x+h_x)} u_{if,jf} + \frac{H_x - h_x}{h_x(H_x+3h_x)} u_{if+1,jf}.
\end{aligned} \tag{23}$$

Substituting (22) into (23) will replace the unknown data point, $u_{ic,jc+\frac{1}{4}}$, with known data points,

$$\begin{aligned}
F_{if-\frac{1}{2},jf} &= -\frac{8h_x}{(H_x+h_x)(H_x+3h_x)} \left(\frac{5}{32} u_{ic,jc+1} + \frac{15}{16} u_{ic,jc} - \frac{3}{32} u_{ic,jc-1} \right) \\
&- \frac{H_x - 3h_x}{h_x(H_x+h_x)} u_{if,jf} + \frac{H_x - h_x}{h_x(H_x+3h_x)} u_{if+1,jf}.
\end{aligned}$$

The final scheme for this approximation is then

$$\begin{aligned}
F_{if-\frac{1}{2},jf} &= -\frac{5}{4} \frac{h_x}{(H_x+h_x)(H_x+3h_x)} u_{ic,jc+1} - \frac{15}{2} \frac{h_x}{(H_x+h_x)(H_x+3h_x)} u_{ic,jc} \\
&+ \frac{3}{4} \frac{h_x}{(H_x+h_x)(H_x+3h_x)} u_{ic,jc-1} - \frac{H_x - 3h_x}{h_x(H_x+h_x)} u_{if,jf} \\
&+ \frac{H_x - h_x}{h_x(H_x+3h_x)} u_{if+1,jf}.
\end{aligned} \tag{24}$$

To determine the order of this scheme, Taylor series expansions of each data point will need to be calculated in terms of the interface point $u_{if-\frac{1}{2},jf}$. These are,

$$\begin{aligned}
u_{ic,jc+1} &= u_{if-\frac{1}{2},jf} - \frac{1}{2} H_x \partial_x u_{if-\frac{1}{2},jf} + \frac{3}{4} H_y \partial_y u_{if-\frac{1}{2},jf} + \frac{1}{8} {H_x}^2 \partial_x^2 u_{if-\frac{1}{2},jf} \\
&- \frac{3}{8} H_x H_y \partial_x \partial_y u_{if-\frac{1}{2},jf} + \frac{9}{32} {H_y}^2 \partial_y^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({H_x}^3) + \mathcal{O}({H_y}^3),
\end{aligned}$$

$$\begin{aligned}
u_{ic,jc} &= u_{if-\frac{1}{2},jf} - \frac{1}{2} H_x \partial_x u_{if-\frac{1}{2},jf} - \frac{1}{4} H_y \partial_y u_{if-\frac{1}{2},jf} + \frac{1}{8} {H_x}^2 \partial_x^2 u_{if-\frac{1}{2},jf} \\
&+ \frac{1}{8} H_x H_y \partial_x \partial_y u_{if-\frac{1}{2},jf} + \frac{1}{32} {H_y}^2 \partial_y^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({H_x}^3) + \mathcal{O}({H_y}^3),
\end{aligned}$$

$$\begin{aligned}
u_{ic,jc-1} &= u_{if-\frac{1}{2},jf} - \frac{1}{2} H_x \partial_x u_{if-\frac{1}{2},jf} - \frac{5}{4} H_y \partial_y u_{if-\frac{1}{2},jf} + \frac{1}{8} {H_x}^2 \partial_x^2 u_{if-\frac{1}{2},jf} \\
&+ \frac{5}{8} H_x H_y \partial_x \partial_y u_{if-\frac{1}{2},jf} + \frac{25}{32} {H_y}^2 \partial_y^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({H_x}^3) + \mathcal{O}({H_y}^3),
\end{aligned}$$

$$u_{if,jf} = u_{if-\frac{1}{2},jf} + \frac{1}{2} h_x \partial_x u_{if-\frac{1}{2},jf} + \frac{1}{8} {h_x}^2 \partial_x^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({h_x}^3),$$

and

$$u_{if+1,jf} = u_{if-\frac{1}{2},jf} + \frac{3}{2} h_x \partial_x u_{if-\frac{1}{2},jf} + \frac{9}{8} {h_x}^2 \partial_x^2 u_{if-\frac{1}{2},jf} + \mathcal{O}({h_x}^3).$$

Substituting these expansions into (24) will result in the following truncation error,

$$F_{if-\frac{1}{2},jf} = \partial_x u_{if-\frac{1}{2},jf+1} + \mathcal{O}\left(\frac{H_x^3 h_x}{(H_x+h_x)(H_x+3h_x)}\right) + \mathcal{O}\left(\frac{H_y^3 h_x}{(H_x+h_x)(H_x+3h_x)}\right).$$

Just as in the linear interpolant case, an estimation for the interface point $(if, jf+\frac{1}{2})$ will also be needed. For the first quadratic interpolant, the three coarse nodes that are adjacent or cornered with the coarse/fine interface will be used,

$$\begin{aligned} L(\tau) &= \sum_{j=0}^{2}\left(u_j \cdot \prod_{f=0,f\neq j}^{2} \frac{\tau-\tau_f}{\tau_j-\tau_f}\right) \\ &= u_{ic,jc+1}\frac{\tau(\tau-H_x)}{2{H_x}^2} - u_{ic+1,jc+1}\frac{(\tau-H_x)(\tau+H_x)}{{H_x}^2} + u_{ic+2,jc+1}\frac{\tau(\tau+H_x)}{2{H_x}^2}, \end{aligned}$$

where $\tau=0$ is associated with the center point, $(ic+1, jc+1)$. Evaluating at $\tau = -\frac{H_x}{4}$ will give the value, $u_{ic+\frac{3}{4},jc+1}$, that is aligned with the fine data

$$u_{ic+\frac{3}{4},jc+1} = L\left(-\frac{H_x}{4}\right) = \frac{5}{32}u_{ic,jc+1} + \frac{15}{16}u_{ic+1,jc+1} - \frac{3}{32}u_{ic+2,jc+1}. \tag{25}$$

Using the three fine data points $u_{ic+\frac{3}{4},jc+1}$, $u_{if,jf}$, and $u_{if,jf-1}$, a quadratic interpolant is created,

$$\begin{aligned} L(\eta) &= \sum_{j=0}^{2}\left(u_j \cdot \prod_{f=0,f\neq j}^{2} \frac{\eta-\eta_f}{\eta_j-\eta_f}\right) \\ &= u_{ic+\frac{3}{4},jc+1}\frac{(2\eta+h_y)(2\eta+3h_y)}{(H_y+h_y)(H_y+3h_y)} - u_{if,jf}\frac{(2\eta-H_y)(2\eta+3h_y)}{2h_y(H_y+h_y)} \\ &+ u_{if,jf-1}\frac{(2\eta-H_y)(2\eta+h_y)}{2h_y(H_y+3h_y)}, \end{aligned}$$

where $\eta=0$ occurs at the coarse/fine interface. Taking the derivative and evaluating at $\eta=0$ yields,

$$\begin{aligned} \partial_y u_{if,jf+\frac{1}{2}} = L'(0) &= \frac{8h_y}{(H_y+h_y)(H_y+3h_y)}u_{ic+\frac{3}{4},jc+1} \\ &+ \frac{H_y-3h_y}{h_y(H_y+h_y)}u_{if,jf} - \frac{H_y-h_y}{h_y(H_y+3H_y)}u_{if,jf-1}. \end{aligned} \tag{26}$$

Combining (25) and (26) produces a scheme in terms of known data points given by

$$\begin{aligned} F_{if,jf+\frac{1}{2}} &= \frac{8h_y}{(H_y+h_y)(H_y+3h_y)}\left(\frac{5}{32}u_{ic,jc+1} + \frac{15}{16}u_{ic+1,jc+1} - \frac{3}{32}u_{ic+2,jc+1}\right) \\ &+ \frac{H_y-3h_y}{h_y(H_y+h_y)}u_{if,jf} - \frac{H_y-h_y}{h_y(H_y+3H_y)}u_{if,jf-1}. \end{aligned}$$

Fully reduced, the scheme is

$$\begin{aligned}
F_{if,jf+\frac{1}{2}} &= \frac{5}{4}\frac{h_y}{(H_y+h_y)(H_y+3h_y)}u_{ic,jc+1} + \frac{15}{2}\frac{h_y}{(H_y+h_y)(H_y+3h_y)}u_{ic+1,jc+1} \\
&- \frac{3}{4}\frac{h_y}{(H_y+h_y)(H_y+3h_y)}u_{ic+2,jc+1} + \frac{H_y-3h_y}{h_y(H_y+h_y)}u_{if,jf} \\
&- \frac{H_y-h_y}{h_y(H_y+3H_y)}u_{if,jf-1}. \qquad (27)
\end{aligned}$$

To determine the accuracy of (27), the Taylor series expansions in terms of $u_{if,jf+\frac{1}{2}}$ will have to be found for each of the five nodes involved. They are

$$\begin{aligned}
u_{ic,jc+1} &= u_{if,jf+\frac{1}{2}} - \frac{3}{4}H_x\partial_x u_{if,jf+\frac{1}{2}} + \frac{1}{2}H_y\partial_y u_{if,jf+\frac{1}{2}} + \frac{9}{32}{H_x}^2\partial_x^2 u_{if,jf+\frac{1}{2}} \\
&- \frac{3}{8}H_xH_y\partial_x\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{8}{H_y}^2\partial_y^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({H_x}^3) + \mathcal{O}({H_y}^3),
\end{aligned}$$

$$\begin{aligned}
u_{ic+1,jc+1} &= u_{if,jf+\frac{1}{2}} + \frac{1}{4}H_x\partial_x u_{if,jf+\frac{1}{2}} + \frac{1}{2}H_y\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{32}{H_x}^2\partial_x^2 u_{if,jf+\frac{1}{2}} \\
&+ \frac{1}{8}H_xH_y\partial_x\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{8}{H_y}^2\partial_y^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({H_x}^3) + \mathcal{O}({H_y}^3),
\end{aligned}$$

$$\begin{aligned}
u_{ic+2,jc+1} &= u_{if,jf+\frac{1}{2}} + \frac{5}{4}H_x\partial_x u_{if,jf+\frac{1}{2}} + \frac{1}{2}H_y\partial_y u_{if,jf+\frac{1}{2}} + \frac{25}{32}{H_x}^2\partial_x^2 u_{if,jf+\frac{1}{2}} \\
&+ \frac{5}{8}H_xH_y\partial_x\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{8}{H_y}^2\partial_y^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({H_x}^3) + \mathcal{O}({H_y}^3),
\end{aligned}$$

$$u_{if,jf} = u_{if,jf+\frac{1}{2}} - \frac{1}{2}H_y\partial_y u_{if,jf+\frac{1}{2}} + \frac{1}{8}{H_y}^2\partial_y^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({h_y}^3),$$

and

$$u_{if,jf-1} = u_{if,jf+\frac{1}{2}} - \frac{3}{2}H_y\partial_y u_{if,jf+\frac{1}{2}} + \frac{9}{8}{H_y}^2\partial_y^2 u_{if,jf+\frac{1}{2}} + \mathcal{O}({h_y}^3).$$

Substituting these expansions into (27) yields,

$$F_{if,jf+\frac{3}{2}} = \partial_y u_{if,jf+\frac{3}{2}} + \mathcal{O}\left(\frac{H_y^3h_y}{(H_y+h_y)(H_y+3h_y)}\right) + \mathcal{O}\left(\frac{H_x^3h_y}{(H_y+h_y)(H_y+3h_y)}\right).$$

Using (24) and (27), the discretization scheme (7) is updated to include only those data points that are known. So,

$$\begin{aligned}
FD_{if,jf} &= \frac{K_{if+\frac{1}{2},jf}(P_{if+1,jf}-P_{if,jf})}{{h_x}^2} \\
&- \frac{K_{if-\frac{1}{2},jf}}{h_x}\left[\frac{8h_x}{(H_x+h_x)(H_x+3h_x)}\left(\frac{5}{32}u_{ic,jc+1}+\frac{15}{16}u_{ic,jc}-\frac{3}{32}u_{ic,jc-1}\right)\right. \\
&- \left.\frac{H_x-3h_x}{h_x(H_x+h_x)}u_{if,jf}+\frac{H_x-h_x}{h_x(H_x+3h_x)}u_{if+1,jf}\right] \\
&+ \frac{K_{if,jf+\frac{1}{2}}}{h_y}\left[\frac{8h_y}{(H_y+h_y)(H_y+3h_y)}\left(\frac{5}{32}u_{ic,jc+1}+\frac{15}{16}u_{ic+1,jc+1}-\frac{3}{32}u_{ic+2,jc+1}\right)\right. \\
&+ \left.\frac{H_y-3h_y}{h_y(H_y+h_y)}u_{if,jf}-\frac{H_y-h_y}{h_y(H_y+3H_y)}u_{if,jf-1}\right] \\
&- \frac{K_{if,jf-\frac{1}{2}}(P_{if,jf}-P_{if,jf-1})}{{h_y}^2},
\end{aligned}$$

and

$$\begin{aligned}
(\nabla\cdot(K(x)\nabla P))_{if,jf} &= FD_{if,jf}+\mathcal{O}\left(\frac{H_x^3}{(H_x+h_x)(H_x+3h_x)}\right)+\mathcal{O}\left(\frac{H_y^3}{(H_y+h_y)(H_y+3h_y)}\right) \\
&+ \mathcal{O}\left(\frac{H_y^3}{(H_x+h_x)(H_x+3h_x)}\right)+\mathcal{O}\left(\frac{H_x^3}{(H_y+h_y)(H_y+3h_y)}\right).
\end{aligned}$$

As in the linear interpolation case, the same unknowns that were replaced with the first derivative approximations are again replaced; this time with the approximation determined by the quadratic interpolants.

2.2.3 Comparison of Linear and Quadratic Interpolation

Quadratic interpolation is an order of magnitude more accurate than linear interpolation. Based on this information alone, quadratic interpolation would seem more desirable; however, this scheme requires a larger support than the linear scheme. In 2D, linear interpolation of the fine cell (i, j) that is adjacent to two coarse cells and two fine cells will use a total of five cells for calculations. In quadratic interpolation, (i, j) would require eight cells (see Figure 4). The larger support of the quadratic interpolation would require more communication in parallel environments.

2.2.4 Intrinsic and Relative Permeability

Intrinsic permeability will be supplied by the user for each cell in the coarsest grid. Daughter cells will inherit the parent's value. At the coarse/fine boundary, harmonic averaging between the coarse and fine cell will be used to determine the instrinsic permeability at the interface. Averaging the coarse grid data with the fine grid data is acceptable in this situation since a ghost fine node would also inherit the coarse cell's data.

At the coarse/fine interface, relative permeability will be calculated at the ghost node using the pressure head found at that node described earlier in the paper. This value as well as the relative permeability from the neighboring fine cell will be used in upwinding to determine the relative permeability at the coarse/fine boundary.

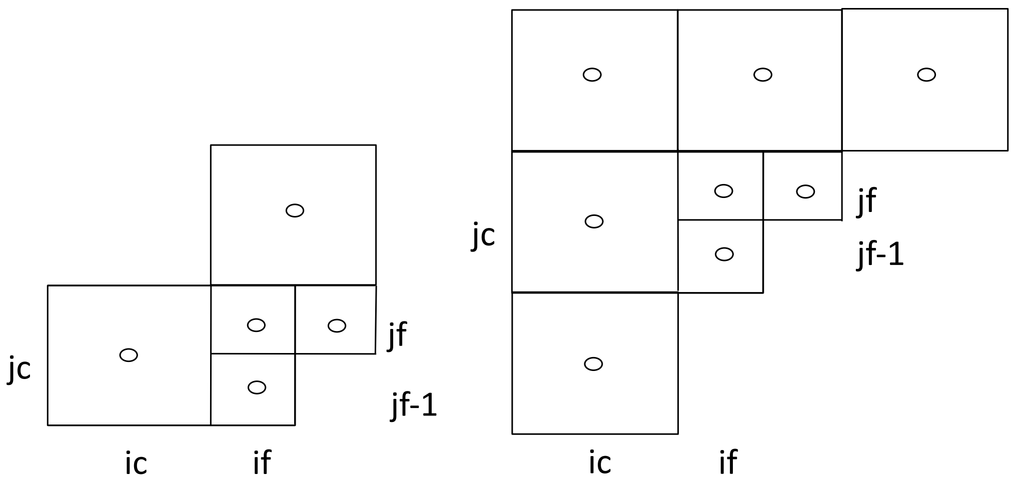

Figure 4: Left: Support for cell (if,jf) consists of 5 cells using linear interpolation. Right: Support for cell (if,jf) consists of 8 cells using quadratic interpolation.

2.2.5 Boundary Conditions

In cell-centered grids, the boundary data will lie on half points, i.e. $(-\frac{1}{2}, j)$. Extrapolation will be used from this data to create a ghost node [1]. If Dirichlet boundary data is given, then a parabola can be created from $u_{-\frac{1}{2},j}$, $u_{0,j}$, and $u_{1,j}$, which can be used to extrapolate the value at the ghost node $(-1, j)$. If Neumann boundary conditions are used, then the parabola will be created through $u_{0,j}$ and $u_{1,j}$ using the given normal derivative at $(-\frac{1}{2}, j)$.

3 Designing and Implementing a Spline/Look-up Capability

ParFlow employs the Van Genuchten model for relative permeability given by

$$k_r(p) = \frac{\left(1 - \frac{(\alpha p)^{n-1}}{(1+(\alpha p)^n)^m}\right)^2}{(1 + (\alpha p)^n)^{m/2}},$$

where $k_r(p)$ is the relative permeability, p is the pressure head of water, α and n are soil parameters, and $m = 1 - 1/n$. Prior to this project, this function was calculated directly. Because of the fractional exponents, these computations could become costly, especially when they were evaluated at every point in a large scale problem. To address this cost, a spline lookup capability was added as an option for this calculation. The remainder of this paper will explain how a spline was applied to the code currently in use for ParFlow. The only source code file that changed was problem_phase_rel_perm.c.

3.1 Setup

To begin, a new struct was created called "VanGTable". In this struct are pointers to five doubles: x, a, d, a_der, and d_der. Three pointers were also added to the Type1 struct (the struct associated with the Van Genuchten calculation of relative permeability): num_sample_points (integer), min_pressure_head (double), and lookup_tables (VanGTable). The pointers in the

VanGTable struct will be used to calculate the spline, which will be explained in a later section. The num_sample_points will be an input from the user telling how many points should be used to create the spline table. The user should have an idea of how many points will be needed to capture the curve accurately enough to solve the problem. The min_pressure_head is also specified by the user; this should be the smallest pressure head used by the problem in a specific region. If a smaller one is used, the relative permeability will be calculated as the relative permeability of the minimum pressure head specified by the user. Note that the user will specify a num_sample_points and a min_pressure_head for each region.

3.2 Functions

Two functions were created within the file: "*VanGComputeTable" and "VanGLookup".

3.2.1 *VanGComputeTable

*VanGComputeTable will return a pointer to a VanGTable. The inputs are the number of interpolation points, the minimum pressure head, α, and n. First, the function allocates a new VanGTable, creating pointers to five arrays set to the size of num_sample_points + 1. After, some local variables are created that will contribute to the spline calculations: h, f, and del are arrays of doubles that represent the distance between successive interpolation points, the Van Genuchten function evaluated at those interpolation points, and the slope of the secant lines between successive points, respectively. Two other arrays, f_der and del_der, are similarly used for the derivative of the Van Genuchten function. The variables $alph$, $beta$, and $magn$ are doubles that are used to ensure that the spline is monotonic (will be explained later). The doubles $opahn$, $ahnm1$, and $coeff$ are used to calculate the Van Genuchten function and its derivative at the specified interpolation points. $interval$ is a double that contributes to the calculation of where the interpolation points will be. Currently, the code uses a fixed width of minimum pressure head divided by the number of sample points + 1. The variable m is set to $1-1/n$. After creating the necessary variables, the code will use the evenly spaced interpolation points to calculate the value of the Van Genuchten function and its derivative at those points, placing them in the arrays a and a_der, respectively, located in the VanGTable.

Now, the monotonic cubic spline will be described [3]. First, a non-monotonic cubic Hermite spline was used. When this failed in the actual code, tests were performed in MatLab that showed that a monotonic spline would do a better job at interpolating the function and its derivative. To create a monotonic spline, several steps must be performed. First, the slopes of the secant lines between successive points must be calculated and placed in the del array

$$del = \frac{f_{i+1} - f_i}{a_{i+1} - a_i},$$

for $i = 0, 1, ..., n-1$. Note that this will be done twice: once for the function and once for the derivative (placed in the del_der array). Next, the tangents at each interpolation point will be set to the average of the secants calculated using that point

$$d = \frac{del_i + del_{i-1}}{2},$$

for $i = 1, 2, ..., n-1$. Note that these are placed in the VanGTable in the d and d_der arrays for the function and the derivative, respectively.

The last step will be to ensure monotonicity. The code will loop through the secant line slopes (del) to check if any are zero. If $del_i = 0$, then $f_{i+1} = f_i$. As a result, the code will set $d_{i+1} = d_i = 0$ to ensure monotonicity. If $del_i \neq 0$, then an additional step will need to be taken. First, two new variables will be defined: $\alpha_i = d_i/del_i$ and $\beta_i = d_{i+1}/del_i$ [10]. To ensure monotonicity, the following condition is needed,

$$\alpha - \frac{(2\alpha + \beta - 3)^2}{3(\alpha + \beta - 2)} > 0.$$

This can be accomplished by restricting the magnitude of (α_i, β_i) to a circle of radius three [10] and modifying the values of d_i and d_{i+1} in the following way:

$$d_i = \frac{3\alpha_i del_i}{\sqrt{\alpha_i^2 + \beta_i^2}},$$

and

$$d_{i+1} = \frac{3\beta_i del_i}{\sqrt{\alpha_i^2 + \beta_i^2}}.$$

After this, the monotonic spline is complete. The function will return the VanGTable containing the location of the interpolation points, the function value at those points, and the average of the secants at successive points (with necessary modifications to ensure monotonicity). The equivalent data for the Van Genuchten derivative will also be included.

3.2.2 VanGLookup

VanGLookup will return the relative permeability calculated using the VanGTable from *VanG-ComputeTable. This function takes as input the pressure head, number of interpolation points, a pointer to the VanGTable, and an integer specifying if the Van Genuchten function or its derivative is needed. First, the function will determine if the pressure head given is below the minimum pressure head. If so, the pressure head is set to the minimum and the relative permeability will be calculated from the spline using the minimum pressure head. If the pressure head is greater than the minimum, then a binary search is performed to find the interval in the VanGTable where the pressure head is located. With the current use of a fixed space interval, a binary search is not needed; however, variably spaced intervals are desired and a binary search will be needed when this is implemented. After the interval is determined, the relative permeability can be calculated using the following function [9]

$$rel\_perm = (2t^3 - 3t^2 + 1)a_i + (t^3 - 2t^2 + t)d_i + (-2t^3 + 3t^2)a_{i+1} + (t^3 - t^2)d_{i+1},$$

where the a_i and d_i come from the corresponding values in the VanGTable. An if statement determines if the value needed is for the Van Genuchten function or its derivative.

This function is called in the section of the code involving "case 1" (the Van Genuchten relative permeability section). If the alphas and ns are given in a file, then there is no need to compute a spline and so the original code will remain. If they are given by region, then VanGLookup will be called and a spline table will be created for each region. If the user specifies the number of interpolation points in the input file to be zero, then the function will continue as it has before: calculating the relative permeability at every point using the Van Genuchten function. A specified number of interpolation points other than zero will result in the spline lookup table being used.

3.3 Preliminary Results

The current modified code has been run on the crater2D problem supplied in the "test" folder in the ParFlow directory. This problem was one of the more difficult ones since the large alphas and small ns resulted in a very steep curve near pressure head equal to zero. Many interpolation points were needed to capture this area when calculating the relative permeability of the original function. However, the derivative was easy to capture with fewer interpolation points. As a result of the difficult parameters, the modified code resulted in no speedup or slowdown in the overall time of the run. These results are promising, however, since several modifications can be made that will ensure speedup: different interpolation points between the function and the derivative and variably spaced interpolation point intervals. In addition, this problem was small compared to the real problems that would be used; the cost of calculating a VanGTable for each region would be amortized over a problem with a much larger size.

3.4 Future Additions

- Vary the spacing between interpolation points for better accuracy where needed.
- Use linear interpolation where a cubic spline is not needed (depends on α and n).
- Use different interpolation points for the function and the derivative.
- Allow the user to input the interpolation points they want the code to use.

References

[1] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, and M.L. Welcome, A Conservative Adaptive Projection Method for the Variable Density Incompressible Navier-Stokes Equations, *J. Comput. Phys.* **142**, 1 (1998).

[2] R.E. Ewing and R. D. Lazarov and P. S. Vassilevski, Local refinement techniques for elliptic problems on cell-centered grids. I: Error analysis, *Math. Comp.*, **56**, pp. 437–461 (1991).

[3] F.N. Fritsch, and R.E. Carlson, Monotone Piecewise Cubic Interpolation, *SIAM J. Num. Anal.* **17**, 2 (1980).

[4] S. J. Kollet and R. M. Maxwell, Integrated surface-groundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model, *Advances in Water Resources*, **29**(7), pp. 945-958, (2006).

[5] S. J. Kollet and R. M. Maxwell, Capturing the influence of groundwater dynamics on land surface processes using an integrated, distributed watershed model, *Water Resources Research*, **44**, doi: 10, 1029/2007WR006004 (2008).

[6] R.M. Maxwell, S.J. Kollet, S.G. Smith, C.S. Woodward, R.D. Falgout, I.M. Ferguson, C. Baldwin, W.J. Bosl, R. Hornung, and S. Ashby, ParFlow User's Manual, Technical Report (2009).

[7] M. Pernice, A Comparison of Strategies for Flux Evaluation at Coarse-Fine Interfaces on Structured Locally Refined Grids, Private Communication.

[8] M. Pernice, and B. Philip, Solution of Equilibrium Radiation Diffusion Problems Using Implicit Adaptive Mesh Refinement, *J. Sci. Comput.* **27**, 1 (2006).

[9] Cubic Hermite spline, Wikipedia Entry: http://en.wikipedia.org/wiki/Cubic_interpolation.

[10] Monotonic cubic interpolation, Wikipedia Entry: http://en.wikipedia.org/wiki/Monotone_cubic_interpolation.