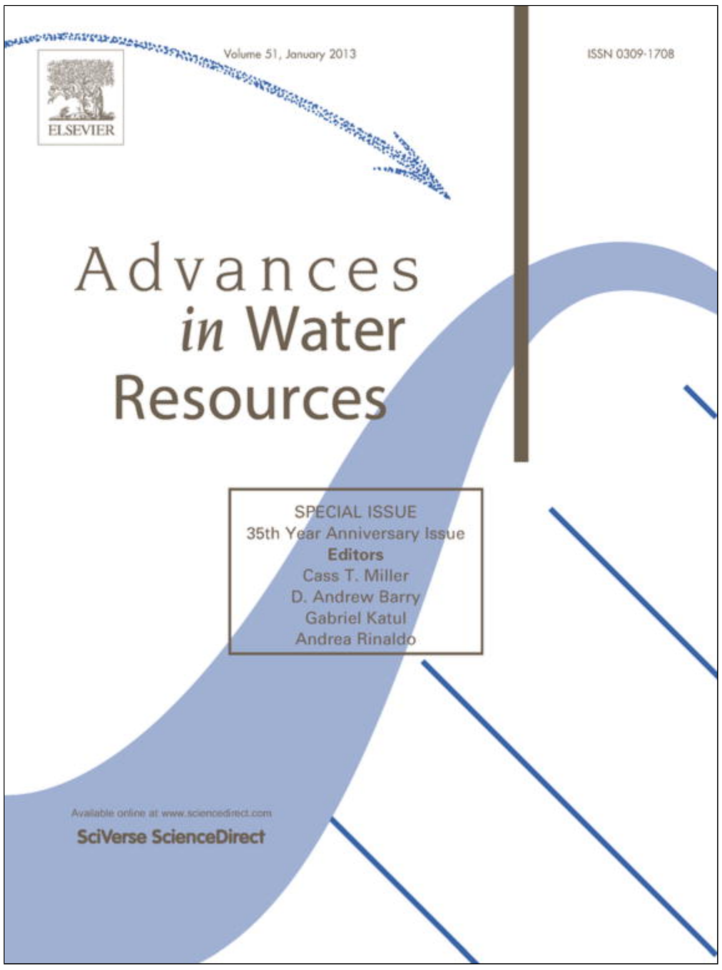
Volume 51, January 2013
ISSN 0309-1708
ELSEVIER
Advances in Water Resources
SPECIAL ISSUE
35th Year Anniversary Issue
Editors
Cass T. Miller
D. Andrew Barry
Gabriel Katul
Andrea Rinaldo
Available online at www.sciencedirect.com
SciVerse ScienceDirect

Advances in Water Resources 51 (2013) 405–437

Contents lists available at SciVerse ScienceDirect

Advances in Water Resources

journal homepage: www.elsevier.com/locate/advwatres

# Numerical simulation of water resources problems: Models, methods, and trends

Cass T. Miller[a,*], Clint N. Dawson[b], Matthew W. Farthing[c], Thomas Y. Hou[d], Jingfang Huang[e], Christopher E. Kees[c], C.T. Kelley[f], Hans Petter Langtangen[g]

[a] *Department of Environmental Sciences and Engineering, University of North Carolina, Chapel Hill, NC 27599-7431, USA*
[b] *Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, TX 78712-0235, USA*
[c] *Coastal and Hydraulics Laboratory, US Army Engineer Research and Development Center, 3909 Halls Ferry Rd., Vicksburg, MS 39180-6133, USA*
[d] *Applied and Computational Mathematics, California Institute of Technology, Pasadena, CA 91125, USA*
[e] *Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599-3250, USA*
[f] *Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205, USA*
[g] *Simula Research Laboratory, P.O. Box 134, NO-1325 Lysaker, Norway*

A R T I C L E   I N F O

*Article history:*
Available online 31 May 2012

*Keywords:*
Adaptive methods
Problem solving environments
Multiscale methods
Integral methods
GPUs
Error estimation and control

A B S T R A C T

Mechanistic modeling of water resources systems is a broad field with abundant challenges. We consider classes of model formulations that are considered routine, the focus of current work, and the foundation of foreseeable work over the coming decade. These model formulations are used to assess the current and evolving state of solution algorithms, discretization methods, nonlinear and linear algebraic solution methods, computational environments, and hardware trends and implications. The goal of this work is to provide guidance to enable modelers of water resources systems to make sensible choices when developing solution methods based upon the current state of knowledge and to focus future collaborative work among water resources scientists, applied mathematicians, and computational scientists on productive areas.

## 1. Introduction

A focus of work appearing in *Advances in Water Resources* since its inception has been on the development of improved mechanistic models of water resources systems and solution approaches to solve such models. Mechanistic models are considered the gold standard for the description of water resources systems and as our fundamental understanding of such systems evolves so too has the formulation of such models. Mechanistic models of water resources systems typically consists of some set of partial differential algebraic equations (PDAEs), which usually are solved approximately using numerical methods. The 35-year history of the journal has documented a remarkable advancement in the sophistication of both the mathematical formulations of mechanistic models of water resources systems and the methods used to approximate the solution of such models.

The water resources field is quite broad in scope, and research of a particular individual, or research group, is often focused on a specific problem—perhaps just an aspect of a single problem. The most interesting and compelling problems in the water resources field are complex and multidisciplinary in nature, often highly so. The range of expertise needed to fundamentally understand, mathematically formulate, and elegantly solve mechanistic representations of advanced applications in water resources at a state-of-the-art level is often beyond the limits of a single researcher or group. Because of the scope and complexity associated with developing a state-of-the-art numerical model of a water resources system, corners are often cut, either because of ignorance on the part of the research team regarding the latest numerical methods or out of a desire for expediency to implement something that gives a solution, even if the approach taken is far from optimally efficient. Scientists might explain such approaches as not being the focus of their work. Motivation aside, the net result of many water resources modeling efforts is an inefficient numerical simulator of a water resources system. Such inefficiency can impact the utility of the model developed and impede the maturation of understanding and application of the approach.

On the other hand, applied mathematicians and computational scientists often find challenging and compelling water resources problems to motivate their work. When this happens, efficient simulators often result that may contribute important advancements in numerical methods. Specialists in numerical methods may not however be focused on the most critical problems of importance to the water resources field or understand subtle features of a

* Corresponding author.
*E-mail addresses:* casey_miller@unc.edu (C.T. Miller), clint@ices.utexas.edu (C.N. Dawson), matthew.w.farthing@usace.army.mil (M.W. Farthing), hou@cms.caltech.edu (T.Y. Hou), huang@email.unc.edu (J. Huang), chris.kees@us.army.mil (C.E. Kees), Tim_Kelley@ncsu.edu (C.T. Kelley), hpl@simula.no (H.P. Langtangen).

0309-1708/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.advwatres.2012.05.008

## Nomenclature

*Roman letters*

| | |
|---|---|
| $A$ | channel cross-sectional area |
| $\mathcal{A}$ | bilinear form |
| $a$ | generic constant |
| $\mathbf{A}$ | coefficient matrix |
| $\hat{A}$ | constitutive coefficient for interface velocity |
| $\mathbf{b}$ | right hand side of stationary iterative method update |
| $C$ | species concentration |
| $c$ | generic constant |
| $D$ | deposition sink |
| $\mathbf{D}$ | dispersion tensor |
| $\mathcal{D}$ | tensor of eddy viscosity coefficients |
| $\hat{\mathbf{D}}$ | second-order tensor related to dispersion (non-dilute TCAT species transport model) |
| $\mathbf{D}_{e,x}$ | eddy viscosity coefficient tensor |
| $\mathbf{D}_{e,y}$ | eddy viscosity coefficient tensor |
| $E$ | erosion source |
| $E^{ms}$ | multiscale map |
| $e$ | solution error |
| $\mathbf{e}$ | unit vector |
| $\mathbf{F}$ | nonlinear residual |
| $\mathbf{F}'$ | Jacobian matrix of $\mathbf{F}$ |
| $f$ | general function |
| $f_c$ | Coriolis parameter |
| $\mathbf{G}$ | Geometric tensor |
| $g$ | gravitational acceleration magnitude |
| $\mathbf{g}$ | gravitational acceleration vector |
| $H$ | hydraulic head |
| $H^1$ | continuous approximation space |
| $H^1_0$ | continuous approximation space of functions with zero trace |
| $\mathcal{H}$ | Heaviside function |
| $h$ | fluid depth |
| $h_e$ | measure of spatial grid length scale |
| $\mathbf{I}$ | identity tensor |
| $\mathbf{I}$ | matrix identity |
| $J^{wn}_w$ | curvature of the $wn$ interface |
| $\mathcal{J}_{c\kappa}$ | index set of phases that can exchange mass with phase $\kappa$ |
| $\mathcal{J}_f$ | index set of fluid phases |
| $\mathcal{J}_f$ | index set of phases |
| $\mathcal{J}_s$ | index set of species |
| $K$ | coarse grid cell |
| $K_{rve}$ | representative volume element |
| $\mathcal{K}$ | Krylov subspace |
| $\mathbf{K}$ | hydraulic conductivity tensor |
| $K_d$ | diffusive wave equation coefficient |
| $\mathbf{k}$ | permeability tensor |
| $\hat{k}$ | interfacial area generation rate parameter |
| $L$ | differential algebraic operator |
| $\mathbf{LU}$ | lower–upper decomposition factorization |
| $\mathbf{M}$ | iteration matrix |
| $\overset{i\kappa\to i\kappa'}{M}$ | mass transfer process for species $i$ from the $\kappa$ phase to the $\kappa'$ phase |
| $MW$ | molecular weight |
| $m$ | diffusive wave flow rate exponent |
| $N$ | computational degrees of freedom |
| $\mathcal{N}$ | wave action density |
| $n$ | generic identifier for number of values |
| $\mathbf{n}$ | unit outer normal |
| $n_e$ | number of elements or cells |
| $n_S$ | number of reacting chemical species |
| $n_s$ | number of split components in operator splitting |
| $n_{sf}$ | number of split components solved over full interval |
| $n_{sh}$ | number of split components solved over half interval |
| $n_{iso}$ | number of components solved in one iterative split operator step |
| $\mathcal{P}_h$ | multiscale finite element space |
| $\mathcal{P}^k$ | complete polynomials of degree less than or equal to $k$ |
| $P$ | number of processors |
| $\mathbf{P}$ | preconditioner |
| $p$ | fluid pressure |
| $p^c$ | capillary pressure |
| $Q$ | flow rate |
| $\mathbf{q}^b$ | bed flux |
| $\mathbf{q}^s$ | suspended sediment flux |
| $R$ | ideal gas constant |
| $\hat{\mathbf{R}}$ | resistance tensor |
| $\hat{R}^d$ | resistance closure relation due to variations in density |
| $\hat{R}^p$ | resistance closure relation due to variations in pressure |
| $r$ | general reaction term |
| $\mathbf{r}$ | residual vector |
| $\mathbf{S_v}$ | momentum sources and sinks |
| $S$ | fluid source term |
| $S_i$ | support of the $i$-the basis function |
| $\mathcal{S}$ | wave action source term |
| $S_s$ | specific storage |
| $S_h$ | volumetric source of water |
| $s$ | saturation |
| $\mathbf{s}$ | Newton increment |
| $s_d$ | DG method selection parameter, $s_d \in \{-1, 0, 1\}$ |
| $\mathcal{T}_h$ | partitioning of a domain |
| $T$ | endpoint of temporal domain |
| $t$ | time |
| $t_i$ | start of solution time interval |
| $u$ | generic dependent variable |
| $\tilde{u}$ | approximate solution in iterative split operator step |
| $u^*$ | exact solution |
| $v_h$ | discrete test function |
| $V_h$ | test space |
| $\mathbf{v}$ | velocity, $(v^x, v^y, v^z)$ |
| $\bar{\mathbf{v}}$ | depth-averaged fluid velocity |
| $\mathbf{v}^{xy}$ | horizontal fluid velocity, $(v^x, v^y)$ |
| $W_h$ | discrete test space |
| $w$ | test function |
| $\mathbf{w}$ | velocity of the interface or common curve |
| $\mathbf{x}$ | spatial coordinate, $(x, y, z)$ |
| $X^{\cdot}$ | partial mass ratio |
| $x^{\cdot}$ | mole fraction |
| $\mathbf{y}$ | generic vector variable |
| $z$ | vertical dimension aligned with gravitational acceleration |
| $z_b$ | bed elevation |
| $\hat{\mathbf{z}}$ | unit vector in the vertical direction |

*Greek letters*

| | |
|---|---|
| $\alpha$ | diffusive wave flow-rate parameter |
| $\alpha_l$ | line-search algorithmic parameter |
| $\gamma$ | interfacial tension |
| $\tilde{\gamma}$ | Cahn–Hilliard infiltration scaling parameter |
| $\hat{\gamma}$ | activity coefficient |
| $\delta_{ij}$ | Kronecker delta function |
| $\tilde{\delta}$ | small number |
| $\epsilon$ | porosity |
| $\epsilon^{\iota}$ | specific entity measure of the $\iota$ entity |
| $\eta$ | linear solver tolerance |
| $\eta_a$ | absolute linear tolerance |
| $\eta_r$ | relative linear tolerance |
| $\theta$ | temperature |

| | |
|---|---|
| $\vartheta$ | wave direction |
| $\kappa$ | wave number |
| $\kappa_c$ | condition number |
| $\kappa_d$ | LDG auxiliary variable |
| $\lambda_l^{\omega_i}$ | $l$-th eigenvalue in $\omega_i$ |
| $\hat{\mu}$ | dynamic viscosity |
| $\xi$ | free surface elevation |
| $\rho$ | mass density |
| $\rho_0$ | reference mass density |
| $\sigma$ | penalty parameter |
| $\boldsymbol{\sigma}$ | stress tensor |
| $\boldsymbol{\tau}$ | viscous and Reynolds stresses |
| $\hat{\tau}_A$ | capillary pressure dynamic coefficient |
| $\tau_a$ | absolute nonlinear tolerance |
| $\tau_r$ | relative nonlinear tolerance |
| $\varphi$ | contact angle |
| $\phi$ | level set function |
| $\phi_j$ | $j$-the multiscale basis function (defined on $K_E$) |
| $\chi_{\kappa s}^{ss}$ | fraction of the solid surface in contact with the $\kappa$ entity, where $\kappa$ is a fluid phase |
| $\Psi$ | pressure head |
| $\psi_j$ | $j$-th multiscale basis function (defined on $K$) |
| $\psi_l^{\omega_i}$ | $l$-th eigenvector in $\omega_i$ |
| $\Omega$ | spatial domain |
| $\Omega_j$ | local element or cell $j$ |
| $\omega$ | mass fraction |

*Subscripts and superscripts*

| | |
|---|---|
| $A$ | species qualifier |
| $B$ | species qualifier |
| eq | equilibrium value |
| $i$ | species qualifier |
| $i$ | generic vector index (subscript) |
| $j$ | generic vector index (subscript) |
| $k$ | iteration index (subscript) |
| $n$ | iteration index (subscript) |
| $t$ | partial derivative with respect to $t$ (subscript) |
| $x$ | partial derivative with respect to $x$ (subscript) |
| up | upwinded value |
| $w$ | phase qualifier |
| $\beta$ | phase qualifier |
| $\kappa$ | phase qualifier |
| $\iota$ | phase qualifier |
| $\pm$ | one-sided limit, $w^{\pm}(x) = \lim_{\epsilon \to 0^{\pm}} w(x)$ |

*Symbols*

| | |
|---|---|
| $^{-}$ | above a superscript refers to a density weighted macro-scale average |
| $^{=}$ | above a superscript refers to a uniquely defined macro-scale average |

| | |
|---|---|
| $[\![\,]\!]$ | jump, $[\![w(x)]\!] = w^{-}(x) - w^{+}(x)$ |
| $\{\}$ | average, $\{w(x)\} = \frac{1}{2}[w^{-}(x) + w^{+}(x)]$ |

*Abbreviations and acronyms*

| | |
|---|---|
| AWR | *Advances in Water Resources* |
| API | application program interface |
| BDF | backward difference formulas |
| Bi-CGSTAB | bi-conjugate gradient stabilized |
| CG | conjugate gradient |
| CGNR | conjugate gradient normal equations |
| CPU | central processing unit |
| DAE | differential algebraic equation |
| DG | discontinuous Galerkin |
| DWE | diffusive wave equation |
| FAS | full approximation schemes |
| FMM | fast multipole method |
| GCS | geological carbon storage |
| GMRES | generalized minimum residual method |
| GPU | graphics processing unit |
| GRK | Gauss Runge–Kutta |
| IE | integral equation |
| IIPG | incomplete interior penalty Galerkin |
| ILU | incomplete lower upper factorization |
| IP | interior penalty |
| JFNK | Jacobian-free Newton Krylov |
| JIT | just in time |
| KDC | Krylov deferred correction |
| KWE | kinematic wave equation |
| LDG | local discontinuous Galerkin |
| LES | large eddy simulation |
| MPI | message passing interface |
| MsFEM | multiscale finite element method |
| MsFVEM | multiscale finite volume element method |
| NIPG | nonsymmetric interior penalty Galerkin |
| ODEs | ordinary differential equations |
| OS | operator splitting |
| PCG | preconditioned conjugate gradient |
| PDAEs | partial differential algebraic equations |
| PDEs | partial differential equations |
| RANS | Reynolds averaged Navier Stokes |
| RE | Richards' equation |
| RKDG | Runge–Kutta discontinuous Galerkin |
| RVE | representative volume element |
| SDC | spectral deferred correction |
| SIPG | symmetric interior penalty Galerkin |
| SSO | sequential split-operator |
| SWE | shallow water equations |
| TCAT | thermodynamically constrained averaging theory |
| TFQMR | transpose free quasi-minimum residual |

mechanistic model. When the focus is entirely on numerical efficiency, there is a tendency to consider simplified model problems and to demonstrate efficiency on problems removed from the class of most compelling applications of the day.

Either of these two common approaches to advancing models of water resources systems has shortcomings and results in either a model that is relatively inefficient from a numerical methods perspective or a numerical model that is insufficiently founded mechanistically to be of optimal benefit to those wishing to apply such a model. While the situation seems somewhat hopeless, steps can be taken to close the gap between the state-of-the-art and the state-of-the-practice and to hasten the rate of advancement of improved methods and approaches. For example, truly collaborative work among scientists, mathematicians, and computational scientists should be encouraged in which no discipline is allowed to cut corners but rather the focus is on elegant, and efficient solutions to the most pressing problems of the day. It is also important to increase understanding of appropriate methods for standard accepted models, and to introduce potentially fruitful methods for currently challenging problems and model formulations for which no current solver exists. It is also worth considering the role of the evolution in computational tools and computer hardware, to ensure that each successive generation of model anticipates future developments rather than rests on aging technology.

The overall goal of this work is to assess the current state of numerical modeling of water resources systems and speculate on potential future trends. The specific objectives of this work are: (1) to highlight an example set of water resources model applications that can drive advancements in computational approaches; (2) to consider algorithmic approaches for approximating mechanistic models; (3) to detail some promising developed and evolving discretization methods; (4) to summarize modern approaches for solving large-scale systems of nonlinear and linear algebraic equations; (5) to review beneficial trends and approaches in computational software environments; and (6) to examine trends in computational hardware and assess the implications these trends may have on the simulation of mechanistic water resources models over the coming decade. These objectives are accomplished in turn in the sections that follow.

## 2. Motivating applications

### 2.1. Overview

As the water resources field evolves, so too do the forms of the mechanistic models of focus to researchers and practitioners. The challenging mechanistic problems of a decade ago are now routine models. The theoretical musings of days gone by are the models of focus today. And, the latest theoretical advancements are fodder for the next generation of models and simulators that will be developed and evaluated over the coming decade and beyond. Such is the nature of scientific advancement. If the situation were different, the scientific field would be stagnant and uninteresting.

Because the focus of this work is on numerical solution approaches to water resources problems, a set of mechanistic models of interest will be summarized to guide the discussion of numerical solution methods of primary interest in this work. The field of water resources is sufficiently broad that only a few illustrative examples of model formulations from subsurface and surface water hydrology will be considered. Furthermore, we will make a distinction between routine models and evolving model formulations. In the former case, guidance will be offered on efficient solution strategies. In the latter case, potential solution approaches and challenges will be considered in subsequent sections.

### 2.2. Subsurface system model formulations

#### 2.2.1. Established models

*Single-phase flow.* Perhaps the most frequently solved groundwater model is a single-phase fluid flow model based upon Darcy's law as an approximate momentum equation and linear, reversible compressibility theory to describe the combined changes in storage effects due to the compressibility of water, the solid media, and the changes in porosity due to rearrangement of the media. The single-phase flow model may be formulated as [78]

$$S_s \frac{\partial H}{\partial t} = \nabla \cdot (\mathbf{K} \cdot \nabla H) + S^w, \tag{1}$$

where $S_s$ is specific storage, $H$ is hydraulic head, $t$ is time, $\mathbf{K}$ is a second-rank symmetric hydraulic conductivity tensor, and $S^w$ is a source term.

Eq. (1) is a linear parabolic PDE, which is elliptic under steady-state conditions. While the solution of this model is trivial in many instances, circumstances exist that can make the solution more challenging. For example, $\mathbf{K}$ usually varies in space, and to a lesser extent $S_s$, and over several orders of magnitude and short spatial scales. While the data typically does not exist to represent this variation precisely, representation of variability in a statistical sense conditioned on known values of $\mathbf{K}$ and evaluated over many realizations can be used to evaluate the effects of heterogeneity and uncertainty, leading to computational challenges, especially as the scale of domain relative to the scale of discretization level becomes large [229,308,360]. The natural desire to discretize finely domains of increasing size leads to discrete problems that will continue to tax even the most advanced high-performance computers for the foreseeable future. Another complicating consideration arises when this model is applied to an unconfined aquifer. Unconfined aquifers are more faithfully represented by a variable saturated flow formulation that accounts for partial drainage and filling of the pore space while representing the multi-fluid nature of the flow system above the water table. An approximate approach is to treat the unconfined system as a nonlinear moving boundary problem, where the top boundary moves as a function of the solution of the model [78]. This approach must approximate the effect of the unsaturated zone above the system being modeled. A third complicating consideration with this standard model is the desire to produce a solution that is locally conservative, which is a property of importance when differentiating the solution to obtain a velocity field for use in a species transport solution [129,280]. The routine nature of the application of this model along with the complicating features that can pose computational challenges and affect numerical method approximation choices combine to make this a model worth considering to ensure good choices are being made in the selection of discretization methods, solvers, and high-performance computing implementation strategies. Over the last decade, solutions of most of the problems above have been proposed by many researchers using a combination of rigorous finite volume or finite difference methods and multiscale approaches, see for example [42,136,209,284].

*Single-phase flow, species transport, and reactions.* Water resources problems that involve chemical composition are often formulated as a combination of a fluid flow problem and some system of equations that describes species transport and reactions. Species transport and reaction problems for saturated flow systems occur routinely, and the combination of biogeochemical reactions and resultant formulations leads to a large set of models in this general class [84,89,256]. Contributing to the cardinality are variations in ways in which mass transfer processes can be represented and the nature of the biogeochemical reactions of concern that can include chemical kinetics, precipitation-dissolution, complexation, enzyme reactions, biological transformations, and particle transport and interactions with solid phases [175,243,244,312]. While many variants of the single-phase flow, species transport, and reaction models exist, many of these models are based upon a general form of the advective–dispersive-reactive equation model that may be given as [73]

$$\frac{\partial(\epsilon C^{iw})}{\partial t} = \nabla \cdot (\epsilon \mathbf{D}^{iw} \cdot \nabla C^{iw}) - \nabla \cdot (\epsilon \mathbf{v}^{iw} C^{iw}) + \epsilon r^{iw} + \sum_{\kappa \in \mathfrak{I}_{cw}} \overset{i\kappa \to iw}{M}, \tag{2}$$

where superscript $i$ is a species qualifier for a macroscale quantity and $w$ is a phase qualifier for a macroscale quantity, $C$ is a species concentration, $\epsilon$ is the porosity, $\mathbf{D}$ is a second-rank dispersion tensor, $\mathbf{v}$ is the fluid velocity vector, $r$ represents all reactions that affect the concentration of species $i$ in the water phase, which is denoted by the superscript $w$, $\overset{i\kappa \to iw}{M}$ represents mass transfer processes from the $\kappa$ phase to the water phase, and $\mathfrak{I}_{cw}$ is the index set of phases that can exchange mass with the water phase, which could include various types of solid phases for the saturated flow case being considered. When mass transfer cannot be assumed to be in equilibrium, a separate conservation of mass equation is required for each type of solid phase in the system [256,268].

The number of species of concern in the model will influence the size of the system needing to be solved. Solid phases that are not in equilibrium with the water phase will also increase the size of the system. If the reactions or mass transfer expression is nonlinear, so too will the resultant model. The nature of the processes being modeled will determine the coupling of the set of transport equations [175,329].

While models of this class are varied and numerous, see for example [296], they are routine in nature. Still, computational challenges remain, such as: (1) many problems are advective dominated, leading to sharp fronts in space and time; (2) multi-species reactive problems can lead to large coupled systems; (3) varying time scales can exist from species to species and for operators within a given species; (4) when spatial heterogeneity is finely resolved in three spatial dimensions, large-scale discrete problems result; and (5) the large number of potential specific model formulations complicates the development of efficient simulators [154,181,184,214,244,309,313,316]. The large number of potential model formulations becomes especially complicated when infiltration and surface water flow are included in so-called integrated hydrologic models [94,299]. While we do not discuss these specific hydrologic formulations further in this paper, the sections below on multiphase flow, surface water, and solution algorithms are directly relevant to integrated hydrologic modeling.

*Richards' equation.* Cases of variably saturated flow occur routinely in water resources systems for applications that range from coupled basin-scale hydrological processes to local-scale drainage and irrigation problems. When the pore space in a porous medium can be occupied by a gas phase and a water phase, a common modeling approach is Richards' equation (RE). RE is a single equation model that results from ignoring the pressure gradient needed to induce flow of the gas phase because of the large contrast that exists between the mobility of the gas and the water phases. RE can be formulated as [337]

$$S_s s^w \frac{\partial \Psi^w}{\partial t} + \frac{\partial \epsilon^w}{\partial t} = \nabla \cdot [\mathbf{K}^w \cdot \nabla(\Psi^w + z)] + S^w, \tag{3}$$

where $s^w = \epsilon^w/\epsilon$ is the saturation of the water phase, $\Psi^w$ is the pressure head of the water phase, $\epsilon^w$ is the volume fraction of the water phase, $\mathbf{K}^w$ is conductivity that depends upon $s^w$, and $z$ is the vertical dimension aligned with the gravitational acceleration vector. Solution of RE requires specification of closure relations of the general form

$$\epsilon^w = \epsilon^w[\Psi^w(t)], \text{ and} \tag{4}$$

$$\mathbf{K}^w = \mathbf{K}^w[s^w(t)]. \tag{5}$$

These general relational forms note a dependence on time, meaning that these relations are hysteretic in nature. Several specific forms of these closure relations typically occur, and they may vary spatially within a given problem [272].

The solution of RE has received considerable attention in the literature. The areas addressed include mass conservative time discretizations [e.g. 97,218], adaptive time discretizations [e.g. 216,218,337,353,354], space discretizations [e.g. 222,234,253, 254,270,271,354], and solvers [e.g. 213,219]. Production simulators are commonplace [e.g. 258,326,352,361]. The considerable attention to the solution of RE supports its importance as a popular hydrologic model of significant importance. This effort also supports the notion that the solution of RE can be computationally challenging and expensive, making the selection of appropriate methods important.

*Multiphase flow.* The simultaneous flow of two, or three, immiscible fluids through porous medium systems is a class of problem that occurs in applications such as drainage and irrigation, as the flow component of problems involving the movement of immiscible fluid contaminants in subsurface systems, and carbon sequestration. The standard model used to model such systems is often written as [272]

$$\frac{\partial(\epsilon^\iota \rho^\iota)}{\partial t} = \nabla \cdot \left[\frac{\rho^\iota \mathbf{k}^\iota}{\hat{\mu}^\iota} \cdot \nabla(p^\iota - \rho^\iota \mathbf{g})\right] \quad \text{for } \iota \in \mathfrak{I}_\mathrm{f}, \tag{6}$$

where $\rho^\iota$ is the density, $\mathbf{k}^\iota$ is the permeability tensor, $\hat{\mu}^\iota$ is the viscosity, $\mathbf{g}$ is the gravitational acceleration vector, $\iota$ is a phase qualifier, and $\mathfrak{I}_\mathrm{f}$ is the index set of fluid phases.

Closure of the model given by Eq. (6) requires specification of the equations of state, for which the isothermal case can be specified generally as

$$\rho^\iota = \rho^\iota(p^\iota, \omega^{i\iota}), \tag{7}$$

where $\omega^{i\iota}$ is the mass fraction of the $i$ species in the $\iota$ phase, and the density depends in general on the mass fraction of all species $i \in \mathfrak{I}_s$ with $\mathfrak{I}_s$ the index set of species. In addition, closure of a multiphase flow model requires relations among fluid pressures, saturations, and permeabilities, which may be specified generally as

$$s^\iota = s^\iota[p^\beta(t)] \quad \text{for } \iota, \beta \in \mathfrak{I}_\mathrm{f} \tag{8}$$

$$\mathbf{k}^\iota = \mathbf{k}^\iota[s^\iota(t)] \quad \text{for } \iota \in \mathfrak{I}_\mathrm{f}, \tag{9}$$

which are in general nonlinear, hysteretic relations. The saturation $s^\iota$ may depend upon capillary pressures between all fluid pairs, which are typically approximated as the difference in volume average fluid pressures. Simplified forms of these closure relations result under certain conditions of wettability.

Multiphase flow modeling using the standard model is routinely applied using established production-level simulators [121,297,352]. The nonlinear and hysteretic nature of the model pose challenges as do desires to model domains in which the model parameters vary over fine spatial scales relative to the domain size of interest, leading to large discrete simulations [74,286]. Commonly used approaches in production codes rely upon fixed grid, low-order methods in space and time [198]. Higher order methods have been shown to result in efficient solution methods for some cases [141,218]. Challenges also exist regarding the solution of problems with discontinuous material properties, which lead to discontinuities at element or cell boundaries in fluid saturations, volume fractions, and permeabilities [145,191,199,300].

*Multiphase flow, species transport, and reactions.* Multiphase flow and species transport and reaction problems in porous medium systems are among the most challenging applications that are commonly performed in the water resources field. The formulation of such models involves a multiphase flow model, of the type outlined above, coupled with a system of species transport and reaction equations that account for changes in the composition of each individual phase. A general form of the transport equations can be written as [272]

$$\frac{\partial(\epsilon^\iota \rho^\iota \omega^{i\iota})}{\partial t} = \nabla \cdot (\epsilon^\iota \rho^\iota \mathbf{D}^{i\iota} \cdot \nabla \omega^{i\iota}) - \nabla \cdot (\epsilon^\iota \rho^\iota \omega^{i\iota} \mathbf{v}^\iota) + \epsilon^\iota r^{i\iota} + \sum_{\kappa \in \mathfrak{I}_{c\iota}} \overset{i\kappa \to i\iota}{M} \quad \text{for } \iota \in \mathfrak{I}_\mathrm{P}, \tag{10}$$

where $\mathfrak{I}_\mathrm{P}$ is the index set of phases.

The situation with multiphase flow and transport processes is similar to the situation with single-fluid phase flow and transport processes. Variations in the number of species, biogeochemical reactions, and interphase mass transfer processes—and the associated sub-models used to represent these features—combine combinatorially to result in a large number of potential model formulations [182]. Due to the importance of multiphase and multicomponent models in petroleum reservoir simulation, there exists rich and highly relevant literature on both models and numerical methods in that community, and we direct the reader to a few canonical texts [59,99,247] and recent articles on advanced numerics for challenging porous media problems [48,53,169]. Several production codes exist that can simulate a set of frequently occurring cases, but many specialty research code variants exist for special cases as well [121,357]. Low-order methods in space and time are standard and fixed grid codes are most commonly used, although some attention has been given to codes that adapt in time or space and time [155,350].

### 2.2.2. Evolving models

Evolving subsurface models include recently formulated models for which mature simulators do not yet exist. Evolving subsurface models also include instances of formulations based upon existing models but which pose special challenges, such as multiscale behavior. The examples included below are intended to illustrate the frontiers of subsurface modeling, but the list is not intended to be exhaustive.

*Cahn–Hilliard infiltration model.* An evolving model in the area of variably saturated flow is an extension of the standard Richards' equation to include higher order terms based on the Cahn–Hilliard phase-field approach [91]. This model was first proposed in [111] and can be written as

$$\frac{\partial(\epsilon s^w)}{\partial t} + \nabla\cdot\left[\mathbf{K}^w(s^w)\cdot\left(\nabla z + \nabla\Psi^w(s^w) + \frac{\tilde{\gamma}}{\rho g}\nabla(\nabla^2 s^w)\right)\right] = 0, \tag{11}$$

where $\Psi^w(s^w)$ is the capillary head-saturation relation, $\tilde{\gamma}$ is a scaling parameter, and $g$ is the magnitude of gravitational acceleration. The last term on the right-hand side results in a fourth-order differential operator that, combined with the well-known nonlinearity in the low-order terms, produces a wetting front instability that is a precondition for reproducing gravity fingering.

*Multiscale geochemical models.* Multiscale geochemical models are of growing interest. One such model involves the precipitation or dissolution of carbonate species. These processes occur when a subsurface geochemical system is out of equilibrium, which can occur for example when supercritical $CO_2$ is injected during a geological carbon storage (GCS) process or through other changes that bring a system out of equilibrium [239]. The injection of an acidic solution lowers the pH of the water, dissolving minerals, and altering the pore morphology and topology and the associated macroscale properties. As the pH changes away from the injection region, precipitation can occur, also changing the pore structure. The net results alters the flow system, perhaps markedly. Such problems are interesting and challenging for a variety of reasons. First, the precipitation or dissolution can occur over very small length scales, much smaller than a typical grid block size in a field-scale simulator. Second, the process can change the local pore morphology and topology and consequently averaged properties such as porosity and permeability. Third, the changes that occur in hydraulic properties can dramatically influence both fluid flow and species transport, making it necessary to resolve these processes accurately in space and time to produce reliable simulations. These factors combine to make multiscale simulation of such processes essential in some cases.

A related example of a multiscale geochemical problem of growing importance is also related to GCS. When $CO_2$ dissolves in a brine the brine becomes more dense [303]. This change in density can lead to gravity fingering. As gravity fingering occurs the $CO_2$ is transported vertically downward, diffuses and disperses, and can locally result in geochemical systems that are out of equilibrium and hence precipitation can occur. The local flow systems are complex and multiscale approaches appear to be the most reliable way of modeling fate and transport processes associated with GCS, although approximate solutions will certainly play a role.

Depending upon the geochemical conditions, the model formulation is typically some subset of the standard models already summarized, either a geochemical model within a single-fluid-phase flow and transport model for the carbonate dissolution/precipitation case, or a multiphase flow and transport formulation for the supercritical $CO_2$ case [292,311]. The challenge comes in resolving the multiple length and time scales involved and evolving the model parameters that depend upon the pore morphology and topology [75,335]. Much work remains to be accomplished before the modeling of such systems become mature.

*Multiscale biological reaction models.* Biological sciences are maturing rapidly and so too are the models used to represent operative biological processes. Standard approaches of representing biological reactions have been based upon kinetic models that represent enzyme utilization, Michaelis–Menten growth models, first-order degradation approximations, or instantaneous rates of reaction [73]. It has long been understood that such kinetic models are simplifications of more complex underlying mechanisms, but the hope has been that the traditional models provide a sufficiently accurate representation of the true behavior to be of use. This assertion is difficult to support in the absence of a more mature understanding of the mechanisms involved.

Recent advances in proteomics and genetics have enabled the development of much more sophisticated, and presumably realistic, simulations of biologically reactive systems. Microbial communities can be sampled, proteins and peptides identified, and genes associated with detailed reaction mechanisms [92]. Such approaches can provide true-to-mechanism details of reaction pathways and promise to deliver more realistic models of biologically mediated reactions [152]. However, with increased mechanistic realism comes increased complexity. It is expected that true-to-mechanism biological models for processes such as uranium reduction involve more than a hundred species and several hundred reactions. The advancement of sophisticated models of this nature will require special considerations for algorithms and data structures and detailed analysis to determine ways in which fully mechanistic models can be represented faithfully, while neglecting unnecessary detail. Such an advancement will require a detailed collaboration of experimentalists, mechanistic modelers, and computational scientists.

*TCAT non-dilute species transport model.* Single-fluid-phase flow and species transport of concentrated species has received recent attention in the literature [347,348]. Traditional Fickian models are not capable of describing the experimentally observed breakthrough profiles, with the observed fronts significantly sharper in space and time than the breakthrough profile for a conservative dilute species.

Approaches to modeling such non-dilute, non-ideal systems have been advanced and are based upon the thermodynamically constrained averaging theory (TCAT) [162]. TCAT is a theoretical framework for deriving models of transport phenomena that are consistent across scales and posed in terms of precisely described variables [161,274]. Specifically, two different approaches have been considered, one in which momentum equations are written for each species in each entity in the system and a second approach in which an entity momentum equation is written and closure relations for deviation velocities are posited based upon an entropy inequality. A simulator has yet to be published and compared to experimental data based upon either approach.

Because the entity-based momentum formulation is simpler, we summarize this model as a suggested target for subsurface modeling and numerical methods advancement. Details on the development of this instance of the general class of model are available in the literature [162].

The non-dilute TCAT species model consists of a conservation of mass equation for the $w$ phase

$$\frac{\partial(\epsilon^w\rho^w)}{\partial t} + \nabla\cdot(\epsilon^w\rho^w\mathbf{v}^{\overline{w}}) = 0, \tag{12}$$

a conservation of momentum equation for the $w$ phase

$$\nabla p^w - \rho^w\mathbf{g}^{\overline{w}} + \epsilon^w(\hat{R}^w - \hat{R}^d)\mathbf{v}^{\overline{w},\overline{s}} + \hat{R}^p\nabla\epsilon^w = 0, \tag{13}$$

a species conservation of mass equation for species $A$ in a two-species $w$ phase in material derivative form

$$\epsilon^w \rho^w \frac{\mathrm{D}^{\overline{w}} \omega^{A\overline{w}}}{\mathrm{D}t} - \nabla \cdot \left\{ \epsilon^w \left( \frac{x^{\overline{\overline{Aw}}} x^{\overline{\overline{Bw}}}}{\omega^{A\overline{w}} \omega^{B\overline{w}}} \right) \left[ \left( X^{\overline{\overline{Aw}}} - \omega^{A\overline{w}} \right) \hat{\mathbf{D}}^{ABwe} \cdot \nabla p^w \right. \right.$$
$$+ \rho^w \frac{R\theta^{\overline{\overline{w}}}}{MW_w} \hat{\mathbf{D}}^{ABwe} \cdot \nabla x^{\overline{\overline{Aw}}} + \rho^w \frac{R\theta^{\overline{\overline{w}}} x^{\overline{\overline{Aw}}}}{MW_w \hat{\gamma}^{Aw}} \hat{\mathbf{D}}^{ABwe} \cdot \nabla \hat{\gamma}^{Aw}$$
$$\left. \left. + \hat{R}^p \left( X^{\overline{\overline{Aw}}} - \omega^{A\overline{w}} \right) \hat{\mathbf{D}}^{ABwe} \cdot \nabla \epsilon^w \right] \right\} = 0, \qquad (14)$$

where

$$\hat{\mathbf{D}}^{ABwe} = \left( \mathbf{I} - \frac{\epsilon^w \hat{R}^d x^{\overline{\overline{Aw}}} x^{\overline{\overline{Bw}}}}{\rho^w \omega^{A\overline{w}} \omega^{B\overline{w}}} \hat{\mathbf{D}}^{ABw} \right)^{-1} \cdot \hat{\mathbf{D}}^{ABw} \qquad (15)$$

and functional forms of the equations of state for the activity coefficient

$$\hat{\gamma}^{Aw} = \hat{\gamma}^{Aw}(p^w, \theta^{\overline{\overline{w}}}, \omega^{A\overline{w}}), \qquad (16)$$

the fluid density

$$\rho^w = \rho^w(p^w, \theta^{\overline{\overline{w}}}, \omega^{A\overline{w}}), \qquad (17)$$

and the fluid viscosity

$$\hat{\mu} = \hat{\mu}(p^w, \theta^{\overline{\overline{w}}}, \omega^{A\overline{w}}) \qquad (18)$$

are required. In addition, complete closure also requires functional forms for $\hat{R}^p$ and $\hat{R}^d$, which while scalars in this example formulation may be tensors more generally. In this model formulation, $\hat{R}^w$ is a resistance coefficient, $\hat{R}^d$ is a closure relation coefficient related to density and a gradient in density, $\hat{R}^p$ is a closure relation related to fluid pressure, $x^{\cdot}$ is a mole fraction, $X^{\cdot}$ is a partial mass ratio, $MW$ is a molecular weight, $\hat{\mathbf{D}}^{ABw}$ is second-order tensor related to dispersion, $\theta$ is the temperature, and $R$ is the ideal gas constant.

*TCAT two-phase flow model.* The TCAT approach has been used to derive a two-fluid-phase flow model that includes a consistent thermodynamic basis that is averaged from the microscale [205], evolution equations for interfacial area and common curve extents based upon averaging theorems to aid in resolving the closure problem [163], and an analysis of capillary pressure terms for systems away from equilibrium, which is derived from the entropy inequality [164]. The TCAT approach results in a potential hierarchy of closed models to describe two-fluid-phase flow through a porous medium system. We consider a simple instance of a model from this hierarchy to illustrate the challenges that lie ahead in modeling such systems.

The system of concern includes two fluid phases and a rigid solid phase under isothermal, non-reactive conditions in the absence of inter-entity mass transfer and where the interfaces and common curve are massless. Under these restrictive conditions, a simple model from the TCAT hierarchy of models can be formulated, which includes mass conservation equations for the fluid phases of the form

$$\frac{\partial(\epsilon^\iota \rho^\iota)}{\partial t} + \nabla \cdot (\epsilon^\iota \rho^\iota \mathbf{v}^{\bar{\iota}}) = 0 \quad \text{for } \iota \in \{w, n\}. \qquad (19)$$

The driving force for flow is the sum of chemical and gravitational potentials. If we approximate this sum of potentials as the sum of a pressure gradient and gravity term then the momentum equations for the fluids can be written as

$$\hat{\mathbf{R}}^\iota_w \cdot \mathbf{v}^{\overline{w}} + \hat{\mathbf{R}}^\iota_n \cdot \mathbf{v}^{\overline{n}} = -\nabla p^\iota + \rho^\iota \mathbf{g} \quad \text{for } \iota \in \{w, n\}, \qquad (20)$$

where the resistance tensors $\hat{\mathbf{R}}^\iota_\kappa$ depend upon the morphology and topology of the fluid distributions, including the measures of fluid saturations and interfacial areas. Equations of state are required and since compositional effects are not considered and the system is isothermal, the general form is

$$\rho^\iota = \rho^\iota(p^\iota) \quad \text{for } \iota \in \{w, n\}. \qquad (21)$$

Eqs. (19)–(21) consist of 10 equations and 13 unknowns. Closure requires conditions for capillary pressure and evolution equations for interfacial areas, which cannot generally be derived from conservation principles. For the case in which pressure differences equilibrate more quickly than interface motion occurs, when interfacial tension is constant, and where fluid pressures averaged over interfaces are approximated by intrinsic volume average pressures, an expression for the rate of approach to capillary pressure equilibrium is

$$\hat{\tau}_A \left[ \epsilon \frac{\partial s^w}{\partial t} + \frac{\gamma^{wn}}{p^n - p^w} \hat{k}^{wn}_1 (\epsilon^{wn} - \epsilon^{wn}_{\mathrm{eq}}) \right] = p^w - p^n + p^c, \qquad (22)$$

where

$$p^c = -\gamma^{wn} J^{wn}_w, \qquad (23)$$

$$\hat{k}^{wn} = \left( \frac{p^c}{p^n - p^w} - 1 \right) \hat{k}^{wn}_1, \qquad (24)$$

$\hat{\tau}_A$ is a capillary pressure dynamic coefficient, $\gamma^{wn}$ is the interfacial tension, and $\hat{k}^{wn}$ is an interfacial area generation rate parameter.

Model closure is complete by specifying an evolution equation for the *wn* interfacial area

$$\frac{\partial \epsilon^{wn}}{\partial t} + \nabla \cdot (\epsilon^{wn} \mathbf{w}^{\overline{\overline{wn}}}) - J^{wn}_w \epsilon \frac{\partial s^w}{\partial t} + \hat{k}^{wn} (\epsilon^{wn} - \epsilon^{wn}_{\mathrm{eq}})$$
$$- \frac{\partial \epsilon^{ws}}{\partial t} \cos \varphi^{\overline{\overline{ws,wn}}} \approx 0, \qquad (25)$$

and functional forms for the curvature of the *wn* interface

$$J^{wn}_w = J^{wn}_w (s^w, \epsilon^{wn}, \epsilon^{ws}), \qquad (26)$$

the wetted fraction of the solid phase at equilibrium

$$\chi^{ss}_{ws} = \chi^{ss}_{ws\mathrm{eq}}(s^w, \epsilon^{wn}), \qquad (27)$$

the wetting fluid–solid interfacial area

$$\epsilon^{ws} = \epsilon^{ss} \chi^{ss}_{ws\mathrm{eq}}(s^w, \epsilon^{wn}), \qquad (28)$$

the velocity of *wn* interface

$$\mathbf{w}^{\overline{\overline{wn}}} = \mathbf{G}^{wn} \cdot (\hat{A}^w \mathbf{v}^{\overline{w}} + \hat{A}^n \mathbf{v}^{\overline{n}}), \qquad (29)$$

and functional forms for the coefficients $\hat{A}^w(\epsilon^{wn}, s^w)$, $\hat{A}^n(\epsilon^{wn}, s^w)$, $\hat{k}^{wn}(\epsilon^{wn}, s^w)$, $\epsilon^{wn}_{\mathrm{eq}}(\epsilon^{wn}, s^w)$, $\cos \varphi^{\overline{\overline{ws,wn}}}$, and $\mathbf{G}^{wn}$. For a system with isotropic interfaces $\mathbf{G}^{wn} = \mathbf{I}/3$.

### 2.3. Surface water model formulations

#### 2.3.1. Established models

Like subsurface flow, there are several established models routinely used to describe surface water hydrodynamics. The most common models are fully two-dimensional or weakly three-dimensional flow models. These models are derived based on several approaches including both depth-averaging (integration) or asymptotic methods using a fixed vertical datum. The common formulation approaches used for this class of models are described in the sections that follow.

*Two-Dimensional Shallow Water Equations.* The two-dimensional shallow water equations (SWE) are the most commonly considered model for surface water currents and free surface elevation. The SWE are derived from the incompressible Navier–Stokes equations under the assumption that the water column is vertically well-mixed and that pressure is hydrostatic, leading to the formulation

$$\frac{\partial p}{\partial z} = -\rho g, \qquad (30)$$

where $\rho$ is water density and $g$ is gravitational acceleration. Integrating the continuity and horizontal momentum equations in the Navier–Stokes equations over the depth of the water column, and applying kinematic boundary conditions at the free surface along with additional assumptions on surface and body stresses, one arrives at the SWE, see [343] for a complete derivation. They can be written in conservative form in two (horizontal) space dimensions as

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\overline{\mathbf{v}}) = S_h, \tag{31}$$

$$\frac{\partial (h\overline{\mathbf{v}})}{\partial t} + \nabla \cdot \left[ h\overline{\mathbf{v}} \otimes \overline{\mathbf{v}} + \frac{1}{2} g h^2 \mathbf{I} \right] = f_c \overline{\mathbf{v}} \times \hat{\mathbf{z}} - gh\nabla z_b + \nabla \cdot \boldsymbol{\tau} + \mathbf{S}_\mathbf{v}, \tag{32}$$

where $h$ is the depth of the flow, $\overline{\mathbf{v}}$ is the depth-averaged velocity, $S_h$ is a volumetric source of water, $z_b$ is the elevation of the bed (bathymetry), $f_c$ is the Coriolis parameter, $\hat{\mathbf{z}}$ is the unit vector in the vertical direction, $\boldsymbol{\tau}$ represents Reynolds (and possibly viscous) stresses, and $S_\mathbf{v}$ is the vector of momentum sources and sinks, which typically includes bottom drag losses and wind/wave gains.

*Diffusive Wave Equation.* The diffusive wave equation (DWE) is a simplification of the SWE that is primarily used in modeling overland flow or flow in wetlands. The model can be derived by neglecting the first two terms in the SWE momentum equation, equating the free surface and bottom gradients ($\nabla h \approx \nabla z_b$), and using an algebraic bottom drag such as the Manning or Chezy formula for $\mathbf{S}_\mathbf{v}$. Algebraic manipulations then lead to a scalar flow equation [257,293]

$$\frac{\partial \xi}{\partial t} = \nabla \cdot [K_d(\xi, \nabla\xi)\nabla\xi], \tag{33}$$

where $\xi = h + z_b$ is the water free surface elevation with respect to some datum and, $K_d$ is a nonlinear function of both $\xi$ and $\nabla\xi$. The DWE shares many similarities with subsurface multiphase flow in both the nonlinearity and possible degeneracy of the spatial operator.

*Kinematic Wave Equation.* The Kinematic Wave Equation (KWE) is the simplest shallow water model and is often used in modeling open channel flow or watersheds under heavy rainfall events. This model neglects the local acceleration, convective acceleration and pressure terms in the momentum equation. For open channel flows, the continuity and momentum equations combine into a single equation of the form [257,293]

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0, \tag{34}$$

$$Q = \alpha A^m, \tag{35}$$

where $A$ is the channel cross-sectional area, $Q$ is the flow rate, and $\alpha$ and $m$ are determined by Manning's or Chezy's friction law.

*Contaminant Transport in Surface Waters.* Transport of contaminants in surface water is governed essentially by a two-dimensional form of Eq. (2) so we do not repeat it here. A range of significant transport processes in surface water are of interest. The specific mathematical formulations and numerical methods are highly dependent on the details of a given transport modeling application. For example, the transport of oil and chemical dispersants following the Deepwater Horizon accident in the Gulf of Mexico was a massive, dynamic contamination scenario affected by wind, waves, currents, and chemical/biological degredation reactions, and a range of Lagrangian and Eulerian transport methods have already been investigated to date [125,260].

*Morphodynamics.* Modeling erosion and deposition is often required in surface water resources and represents a distinct form of the coupled flow and transport. We consider

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\overline{\mathbf{v}}) = S_h, \tag{36}$$

$$\frac{\partial (h\overline{\mathbf{v}})}{\partial t} + \nabla \cdot \left[ h\overline{\mathbf{v}} \otimes \overline{\mathbf{v}} + \frac{1}{2} g h^2 \mathbf{I} \right] = -gh\nabla z_b + \mathbf{S}_\mathbf{v}, \tag{37}$$

$$\frac{\partial (h\epsilon^s)}{\partial t} + \nabla \cdot \mathbf{q}^s = E - D, \tag{38}$$

$$\frac{\partial (z_b\epsilon^b)}{\partial t} + \nabla \cdot \mathbf{q}^b = D - E, \tag{39}$$

where in addition to the water depth $h$, we have the suspended sediment volume fraction $\epsilon^s$, suspended sediment flux $\mathbf{q}^s$, erosion source $E$, deposition sink $D$, bed volume fraction constant $\epsilon^b$, and bed flux $\mathbf{q}^b$. Note that the Reynolds and viscous stress, and Coriolis terms have been neglected in Eq. (37) for simplicity. See for example [277] for similar formulations and numerical simulations.

*Wave Action.* Surface wind-waves can impart significant wave radiation stresses in the SWE momentum equations. In turn, the currents and depths computed by the SWE can affect the transport of surface waves. Thus wave-current interaction can be important in many coastal ocean problems, especially during hurricane events.

In the coastal ocean, away from the surf zone, a governing equation for wind waves can be derived as a wave action density balance, which is a hyperbolic balance law in a five dimensional domain where the coordinate directions can be taken as time ($t$), space ($x$ and $y$), wave number ($\kappa$), and wave direction ($\vartheta$). The initial-boundary value problem can be described as: find the wave action density $\mathcal{N}(t, \mathbf{x}, \kappa, \vartheta)$ satisfying

$$\frac{\partial \mathcal{N}}{\partial t} + \nabla \cdot (\mathbf{v}\mathcal{N}) + \frac{\partial (v^\kappa \mathcal{N})}{\partial \kappa} + \frac{\partial (v^\vartheta \mathcal{N})}{\partial \vartheta} = \mathcal{S}, \tag{40}$$

where $\mathcal{S}(t, \mathbf{x}, \kappa, \vartheta, \mathcal{N})$ is the sum of wave action sources (potentially nonlinear in $\mathcal{N}$), and $\mathbf{v}(t, \mathbf{x}, \kappa, \vartheta)$ is the five-dimensional propagation velocity, which does not depend on $\mathcal{N}$. The left-hand side of Eq. (40) is a linear wave equation with spatially variable velocity, which lends itself to specialized numerical methods. Furthermore, the domain is the tensor product of a cube $([0,T] \times [\kappa_{\min}, \kappa_{\max}] \times [\vartheta_{\min}, \vartheta_{\max}])$ with a potentially irregular two-dimensional spatial domain $\Omega$. See [206,242] for derivations.

*Three-Dimensional Shallow Water Equations.* Three-dimensional shallow water (3D-SWE) models can be derived for cases where there is a three-dimensional flow structure, due for example to density or eddy viscosity variations, but the hydrostatic pressure relation still holds. There are a number of ways to formulate the 3D-SWE equations. Here, we follow [119] and write the fluid continuity and momentum equations as

$$\nabla \cdot \mathbf{v} = 0, \tag{41}$$

$$\frac{\partial \mathbf{v}^{xy}}{\partial t} + \nabla \cdot (\mathbf{v} \otimes \mathbf{v}^{xy} - \mathcal{D}\nabla\mathbf{v}^{xy}) = f_c \mathbf{v}^{xy} \times \hat{\mathbf{z}} - g\nabla_{xy}\xi + \mathbf{S}_\mathbf{v} - \frac{1}{\rho_0}\nabla_{xy}p^a - \frac{g}{\rho_0}\int_z^\xi \nabla_{xy}\rho \, \mathrm{d}z, \tag{42}$$

where $\nabla_{xy} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$, $\mathbf{v} = (v^x, v^y, v^z)$ is no longer depth-averaged, $\mathbf{v}^{xy} = (v^x, v^y)$ is the horizontal velocity, $\rho_0$ is a reference density, $p^a$ is the atmospheric pressure, and density variations outside gravitational terms have been neglected (Boussinesq approximation). The $\mathcal{D}\nabla\mathbf{v}^{xy}$ term represents a turbulence closure model where the tensor of eddy viscosity coefficients $\mathcal{D}$ is block $2 \times 2$ with positive semi-definite $3 \times 3$ matrices along the diagonal. The tensor of eddy viscosity coefficients is given by

$$\mathcal{D} = \begin{pmatrix} \mathbf{D}_{e,x} & 0 \\ 0 & \mathbf{D}_{e,y} \end{pmatrix}, \tag{43}$$

where $\mathbf{D}_{e,x}$ and $\mathbf{D}_{e,y}$ are $3 \times 3$ positive semi-definite matrices [119]. In addition to Eqs. (41) and (42), either the free surface kinematic boundary condition or the depth-integrated continuity equation

$$\frac{\partial \xi}{\partial t} + \frac{\partial}{\partial x}\int_{z_b}^{\xi} v^x \mathrm{d}z + \frac{\partial}{\partial y}\int_{z_b}^{\xi} v^y \mathrm{d}z = 0 \tag{44}$$

is typically used to resolve the free surface elevation, $\xi$.

### 2.3.2. Evolving models

The transition from established to evolving surface water formulations is spanned by a class of free surface hydrodynamic models that dispense with the hydrostatic assumption, Eq. (30), and take the two-fluid phase formulation as their point of departure. This class of models has been studied widely over the last 30 years, proceeding from early work on the basic volume-of-fluid and level set methods for regular grids [189,298,304,318,334] to more recent hybrid and unstructured methods [282,285,287,332,338]. These approaches are now considered standard in some contexts. On the other hand, the use of such formulations for water resources investigations has been less common, and developing efficient numerical methods for complex environmental flows remains a challenge [220,336]. To be more specific, these formulations are based on the assumption of incompressible, Newtonian flow of both the air and water phase

$$\nabla\cdot\mathbf{v}^{\iota}=0, \tag{45}$$

$$\frac{\partial(\rho^{\iota}\mathbf{v}^{\iota})}{\partial t}+\nabla\cdot(\rho^{\iota}\mathbf{v}^{\iota}\otimes\mathbf{v}^{\iota}-\boldsymbol{\sigma}^{\iota})=S^{\iota}_{\mathbf{v}}, \tag{46}$$

$$-p^{\iota}\mathbf{I}+2\hat{\mu}^{\iota}\nabla^{s}\mathbf{v}^{\iota}=\boldsymbol{\sigma}^{\iota},\quad \iota\in\{w,a\}, \tag{47}$$

and a sharp fluid–fluid interface across which the boundary conditions

$$(\mathbf{v}^{w}-\mathbf{v}^{a})\cdot\mathbf{n}^{wa}=0,\ \text{and} \tag{48}$$

$$(\boldsymbol{\sigma}^{w}-\boldsymbol{\sigma}^{a})\cdot\mathbf{n}^{wa}=\mathbf{f}^{wa} \tag{49}$$

hold. Here, $\boldsymbol{\sigma}$ is the stress tensor, $\nabla^{s}$ is the symmetric gradient, and $\mathbf{n}^{wa}$ is the interface unit normal. No mass is assumed to cross the fluid–fluid interface in Eq. (49), and $\mathbf{f}^{wa}$ represents interfacial forces due to surface tension [318]. Depending on the application, Eqs. (45)–(49) can be simplified by neglecting air-phase dynamics and assuming constant $p^{a}$. For turbulent flows, various closure relations can be introduced to Eqs. (46) and (47) with possibly additional evolution equations (e.g. $k$–$\epsilon$ or $k$–$\omega$ Reynolds averaged Navier Stokes (RANS) formulations [201,259]).

*3D Air/Water.* Eqs. (45)–(49) require accounting for the evolution of the air–water interface. Different choices lead to level set, volume of fluid, interface tracking, or even hybrid approaches [139,220,334,340]. For instance, a minimal level set formulation can be written

$$\nabla\cdot\mathbf{v}=0, \tag{50}$$

$$\frac{\partial(\rho\mathbf{v})}{\partial t}+\nabla\cdot(\rho\mathbf{v}\otimes\mathbf{v}-2\hat{\mu}\nabla^{s}\mathbf{v})=-\nabla p+\rho\mathbf{S}_{\mathbf{v}}, \tag{51}$$

$$\frac{\partial\phi}{\partial t}+\mathbf{v}\cdot\nabla\phi=0, \tag{52}$$

$$[1-\mathcal{H}(\phi)]\rho^{a}+\mathcal{H}(\phi)\rho^{w}=\rho, \tag{53}$$

$$[1-\mathcal{H}(\phi)]\hat{\mu}^{a}+\mathcal{H}(\phi)\hat{\mu}^{w}=\hat{\mu}, \tag{54}$$

where the fluid–fluid interface is described by the zero contour of $\phi$, $\mathcal{H}$ is the Heaviside function, $\mathbf{f}^{wa}=0$, and $\mathbf{n}^{wa}=\nabla\phi/\|\nabla\phi\|$ [322,333].

*3D Morphodynamics.* A promising approach for modeling sediment dynamics is the use of mixture theory to derive fully three-dimensional models of turbulent, incompressible free surface flow, including sediment phase mass and momentum balances. We present a slightly simplified formulation of a model recently studied for coastal sand dynamics [63–65]:

$$\frac{\partial(\rho^{f}\epsilon^{f})}{\partial t}+\nabla\cdot(\rho^{f}\epsilon^{f}\mathbf{v}^{f})=0, \tag{55}$$

$$\frac{\partial(\rho^{f}\epsilon^{f}\mathbf{v}^{f})}{\partial t}+\nabla\cdot[\rho^{f}\epsilon^{f}\mathbf{v}^{f}\otimes\mathbf{v}^{f}-\varepsilon^{f}\boldsymbol{\sigma}^{f}]=\rho^{f}\epsilon^{f}\mathbf{g}-\mathbf{f}^{fs}, \tag{56}$$

$$\frac{\partial(\rho^{s}\epsilon^{s})}{\partial t}+\nabla\cdot\rho^{s}\epsilon^{s}\mathbf{v}^{s}=0, \tag{57}$$

$$\frac{\partial(\rho^{s}\epsilon^{s}\mathbf{v}^{s})}{\partial t}+\nabla\cdot[\rho^{s}\epsilon^{s}\mathbf{v}^{s}\otimes\mathbf{v}^{s}-\epsilon^{s}\boldsymbol{\sigma}^{s}]=\rho^{s}\epsilon^{s}\mathbf{g}+\mathbf{f}^{fs}, \tag{58}$$

$$\frac{\partial\phi^{aw}}{\partial t}+\epsilon^{f}\mathbf{v}^{f}\cdot\nabla\phi^{aw}=0, \tag{59}$$

where $\rho^{f}$ and $\rho^{s}$ are the fluid and sediment densities, $\epsilon^{f}$ and $\epsilon^{s}$ are the fluid and sediment volume fractions $(\epsilon^{f}+\epsilon^{s}=1)$, $\mathbf{v}^{f}$ and $\mathbf{v}^{s}$ are the fluid and sediment velocities, $\boldsymbol{\sigma}^{f}$ and $\boldsymbol{\sigma}^{s}$ are the fluid and sediment stress tensors, and $\mathbf{f}^{fs}$ is the fluid-to-sediment interphase momentum exchange. Model formulations [63–65] also include a $(k-\epsilon)$ turbulence closure model, which accounts for sediment effects (and introduces two additional evolution equations for the turbulent kinetic energy and its rate of dissipation).

Unsteady and fully turbulent conditions are widely encountered in surface water flows (and even in some subsurface problems like flows in fractured media). Several of the models listed above include standard terms for parameterizing turbulence effects. This is far from a settled issue, however. In environmental applications it is often the transport of dissolved or suspended components that is the ultimate goal of modeling, and transport in flow regimes from transitional to fully turbulent conditions continues to be a challenge from both a fundamental modeling and numerical approximation perspective. Indeed, the past decade has seen particular growth in the theory and usage of large eddy simulation (LES) approaches and turbulence closure relations for RANS equations, and such issues are considered an important, evolving research topic across many disciplines [77,190,278].

## 3. Solution algorithms

The models outlined in Section 2 represent systems of PDAEs, which may contain elliptic, parabolic, or hyperbolic components, and these models may be either linear or nonlinear in nature. For example, compositional problems include components to determine the velocity field of the phases present and components to determine the makeup of each phase, which in turn involves advective, dispersive, and reactive operators. The range and mathematical character of these models is broad. The development of sound solution approaches requires consideration of the algorithm to be used to compute the approximate solution, the most appropriate choice of which in turn is influenced by the particular model formulation of interest.

By solution algorithm, we mean the basic approach taken to resolve coupling in a given PDAE system. It represents decomposition of the solution of the full system into component parts (or subsystems) and the means for recovering the overall solution from the individual parts. This algorithm is independent of the spatial and temporal discretization techniques used to approximate the component subsystems, and the methods used to solve the resultant nonlinear or linear algebraic systems that arise from a discretization procedure. Individual components may also be subject to decomposition or splitting, since they can be PDAE systems themselves, and algorithmic considerations in the broader sense certainly exist within the subsystems as well (e.g. adaption algorithms used in temporal or spatial approximations or multigrid solution of linear systems). These component-wise considerations are discussed in the sections that follow. Here we focus on the overall solution algorithm for the full PDAE system.

Standard solution algorithms either treat a model as one fully coupled system at each discrete time step, or follow a natural sequential decoupling of the model. Fully coupled solves can be linear or nonlinear in nature depending upon the PDAE model, and the solution can be of high dimension when a domain is finely discretized and many unknowns exist at each discrete grid point. This solution of large coupled systems can be expensive and challenging to solve efficiently in parallel.

Natural sequential decoupling can be applied, for example, if the flow field is changing on a different time scale than the one on which transport and reaction processes operate. A limit of this notion is when the flow field is steady state or quasi-steady state.

Sequential decoupling is intuitive when the coupling is one sided; for example, when the transport and reaction system depends upon the flow solution, but the flow solution does not depend upon the transport and reaction solve. Such one-sided coupling occurs physically when the density and viscosity are not compositionally dependent, and in the case of multiphase porous medium systems when the volume fractions are not affected by transport and reaction. Sequential decoupling can also be applied even with systems that are formally coupled in both directions, but a separation of time scales occurs. An example of such a case is the dissolution of a trapped organic phase into water in a porous medium system [202,273]. Similar notions apply to other multiphase mass transfer problems in porous medium systems, such as problems that involve precipitation and dissolution.

More complex models of subsurface and surface water flow and transport contain a sufficient number of unknowns and/or a disparate set of operators, which are not amenable to the standard algorithms discussed above. In this case, operator splitting (OS) approaches are of use. We broadly define OS as the solution of a model by a sequence of component solves with each component consisting of some subset of the operators in the original model. The splitting may decompose the model based upon complete equations that are a subset of the entire model or some subset of the operators within an individual equation. The splitting may also decompose an operator into components using, for example, subsets of the dimensions of a spatial operator, or into scales in a multiscale method, which is a class of method described in detail below. OS may include an arbitrary number of steps, and it may be iterative or non-iterative in nature.

OS methods are well-documented in the applied mathematics and water resources literature and a topic of sustained and continuing interest [e.g. 70–72,80,95,146,147,214,325]. There are three primary motivations for the use of OS methods. First, OS can result in smaller component solves. If the component solves scale less favorably than linearly in the number of unknowns, then dividing the solution into components can reduce the computational expense of the solution, substantially in some cases. Second, OS methods can subdivide a problem into pieces that are each naturally suited to a particular solution method. For example, an advective operator can be approximated by a high-resolution numerical method developed for hyperbolic conservation laws, while a reaction solve can be solved efficiently by a DAE solver for stiff systems of equations. Such approaches have merit because each of the component solves can be done with high efficiency by methods that are well-suited to the component being solved. Third, OS methods are often used to allow component solves to be processed in parallel to take advantage of modern high-performance computing systems. While such approaches can lead to simple parallel algorithms, load balancing issues can arise in naive implementations.

Consider a general model formulation of the form

$$\mathbf{A}\mathbf{u}_t = \mathbf{L}(\mathbf{u}) \quad \text{for } t \in [t_i, t_i + \Delta t], \tag{60}$$

where $\mathbf{A}$ is coefficient matrix, $\mathbf{u}$ is a vector of dependent variables, $\mathbf{L}$ denotes a vector of differential–algebraic operators, the subscript $t$ denotes temporal differentiation, $t_i$ is an initial time, and $\Delta t$ is a time step. Expressing water resources models of interest in this form abstracts away the physical details and allows for a focus on algorithmic approaches.

OS algorithms can be characterized by the order of the truncation error that they introduce and the means used to control this truncation error. The sequential split-operator (SSO) method is common and can be written as

$$\mathbf{A}\mathbf{u}_t = \mathbf{L}_i(\mathbf{u}) \quad \text{for } i = 1, \ldots, n_s; \; t \in [t_i, t_i + \Delta t]; \tag{61}$$

where

$$\mathbf{u}_i(t_i) = \mathbf{u}_{i-1}(t_i + \Delta t) \quad \text{for } i > 1, \tag{62}$$

$n_s$ is the number of split components in the algorithm, and $\mathbf{u}_i$ denotes the solution vector from the $i$th component solve. In Eq. (61), a given equation from the vector system is advanced a time step $\Delta t$ in component solves. Because all components are not approximated simultaneously, an OS error of order $(\Delta t)$ is introduced. The only means to control this error is through a reduction in the OS step $\Delta t$. Naive SSO algorithms neither estimate nor control the OS error introduced into the solution as a formal part of the algorithm.

OS accuracy can be improved through the use of a Strang strategy [330], which can be summarized as

$$\begin{aligned}
\mathbf{A}\mathbf{u}_t &= \mathbf{L}_i(\mathbf{u}) \quad \text{for } i = 1, \ldots, n_{sh}; \; t \in [t_i, t_i + \Delta t/2],\\
\mathbf{A}\mathbf{u}_t &= \mathbf{L}_i(\mathbf{u}) \quad \text{for } i = n_{sh} + 1, \ldots, n_{sh} + n_{sf}; \; t \in [t_i, t_i + \Delta t],\\
\mathbf{A}\mathbf{u}_t &= \mathbf{L}_i(\mathbf{u}) \quad \text{for } i = 1, \ldots, n_{sh}; \; t \in [t_i + \Delta t/2, t_i + \Delta t],
\end{aligned} \tag{63}$$

where $n_{sh}$ is the number of operators that are computed over a half time step, $n_{sf}$ is the number of operators that are computed over a full time step, and the initial conditions for all component solves are assigned based upon the solution from the previous component solve, similar to the strategy given by Eq. (62). Strang splitting can reduce the splitting to order $(\Delta t)^2$ in certain cases with additional expense compared to the SSO approach [157]. Alternating the order of splitting to advance a time step can reduce the performance penalties for Strang splitting and lead to an equivalent order of accuracy [275,341]. Neither Strang splitting nor the alternating OS method is typically implemented to either estimate or control OS error.

Another algorithm that has received some attention in recent years is the iterative split-operator (ISO) algorithm [158,214], which may be summarized as

$$\mathbf{A}\mathbf{u}_t = \mathbf{L}(\tilde{\mathbf{u}}_i) \quad \text{for } i = 1, \ldots, n_{iso}; \; t \in [t_i, t_i + \Delta t], \tag{64}$$

where $\tilde{\mathbf{u}}_i$ denotes a vector that includes some values of $\mathbf{u}$ that are solved for over a time step and other entries of $\mathbf{u}$ that are approximated, and $n_{iso}$ is the number of ISO component solves in one iteration of the algorithm. The ISO algorithm can be iterated until suitable convergence is obtained. It has been proven, subject to sufficiently accurate component solves, that the ISO method converges quadratically [214]. Open issues exist for the ISO algorithm, including conditions under which a given splitting approach convergences and how to best preserve the quadratic convergence rate.

As mentioned above, relatively little has been done for temporal error control and adaption within the context of OS methods [73], although there have been a few recent efforts to develop adaptive approaches for estimating and controlling error due to operator splitting [146,147,157]. Specifically, the approaches from [146,147] extend residual-based *a posteriori* techniques [79,148] to control splitting error. These approaches are elegant and promising but do require solution of a linearized dual problem. Moreover, unlike more traditional elliptic and parabolic PDEs for which they have been quite successful [142,143,301], the adjoint for a split formulation can differ significantly from the adjoint of the fully coupled problem and take care to derive [146]. The work in [157] takes a different path and estimates splitting error using either variable order splittings or step doubling in a Richardson extrapolation approach. Even though it is simpler and less tailored to the operators in a given splitting than adjoint-based techniques, the resulting method is relatively straightforward and generic. In either case, the resulting error estimate can in turn be used to ensure that a specified error criterion has been met and to estimate the next time step size to meet a user-specified error tolerance.

In short, fully coupled and OS solution procedures of long standing continue to be relevant today. In our view, the models solved by the water resources community are growing in complexity, and so the tradeoffs between fully coupled and OS solutions will also continue. Increased computational resources make fully coupled solutions to problems of the size common in the past more tenable. On the other hand, the desire to solve still more complex (and larger) problems means that there will continue to be a demand for OS techniques. The need for improved OS methods with error control and adaption seems clear, whether for established or evolving models.

## 4. Evolving discretization methods

A wide variety of discrete approximation methods for the PDAE systems outlined in Section 2 exist. Since many of these methods are well established, and well documented elsewhere, a review of routine discretization methods is not necessary. Rather, the focus of this section is on a set of evolving methods that are especially promising but which are not yet routinely used in production-level simulators of water resources systems. Applications of these evolving discretization methods to established models could result in more efficient simulators than those currently used for production simulations. These evolving discretization methods are also applicable to the evolving water resources models summarized in Section 2.

The unifying theme among these evolving discretization techniques is based upon three goals: subscale resolution, qualitative correctness, and *a posteriori* error control. Under resolution is a continuing challenge in modeling of water resources systems. Water resource modeling is replete with problems where solutions at the spatial and temporal scales of interest are either controlled or heavily influenced by physical processes and solution dynamics at a much smaller scale. Unfortunately, resolving these small-scale processes using a direct approximation at the small scale is usually impractical if not impossible. Responding to this challenge is a central goal of multiscale methods, but even high-resolution advection schemes can be seen as efforts to approximate steep transitions accurately without recourse to underlying boundary layers.

The idea of monotonicity in scalar hyperbolic PDEs leads to the second theme, which is qualitative correctness. By this we mean that there are solution properties beyond absolute measures of truncation error that a discretization scheme should honor. Besides monotonicity, where appropriate, clear examples include global and local mass conservation, particularly as related to velocity fields in transport problems.

A third theme of importance is the control of discretization error. Most water resources simulators currently in use neither measure nor control error resulting from discretization methods. As numerical simulation methods continue to evolve, explicit control of error will become an increasingly desirable feature. The importance of error control was discussed in Section 3, and it is clearly important when considering discretization methods. Unfortunately, providing an adequate review of relevant *a posteriori* error estimation methods is beyond our scope here. Rather, we refer readers to existing reviews [79] and example applications in water resources [76,238,246,254,328,339,345]. We also note that recent efforts have brought residual-based error estimation techniques [142,143,301] to bear on nonlinear multiphysics problems [146–148].

### *4.1. Discontinuous Galerkin methods*

Discontinuous Galerkin (DG) methods have been developed extensively over the past two decades for a wide variety of problems in water resources and many other applications. Recent publications that describe DG methods in significant detail are [105,186,305]. As opposed to the more standard continuous Galerkin finite element method, which uses $H^1$ approximating spaces, the DG method allows the freedom of using completely discontinuous approximating spaces; i.e., polynomials defined on each element without any requirements on continuity at element boundaries. Along faces between elements, quantities such as advective and diffusive fluxes are approximated by so-called "numerical fluxes," which may be based on Riemann solvers in the case of nonlinear hyperbolic problems, or may be as simple as averages or one-sided values. In addition, jumps in the solution across element faces are controlled through "penalty" terms, which must be judiciously added to make the methods well-defined and stable.

The advantages of the DG method over continuous Galerkin methods are the ability: (1) to preserve local conservation properties, such as conservation of mass; (2) to easily refine the mesh locally within an element without the difficulty of dealing with hanging nodes ($h$ adaptivity); (3) to use different polynomials on each element ($p$ adaptivity) depending on the smoothness of the problem; (4) to treat boundary and other external conditions weakly; and (5) to produce efficient parallel implementations, as most of the work is done at the element level and the stencil usually involves only neighboring elements.

The primary disadvantage of the DG method is that it can be significantly more computationally expensive than continuous Galerkin methods in terms of the number of degrees of freedom per variable on a given mesh. A fairer comparison, however, may be computational effort vs. accuracy. For example, in Kubatko et al. [245], a comparison is made between DG and continuous Galerkin methods for the shallow water equations on a computational effort vs. accuracy basis, and there are numerous cases where the DG method outperforms its continuous Galerkin counterpart. There are also applications when preserving local conservation properties is essential, such as when coupling flow and transport [118], and here DG methods have clear advantages. It is also possible to mix discontinuous and continuous approximating spaces in various ways, again depending on the application, see [49,120] for discussion of so-called hybrid approaches.

DG methods appear to have been first introduced in the 1970's for stationary neutron transport problems by Reed and Hill [302]. Lesaint and Raviart analyzed this method and named it the "discontinuous Galerkin" method [252]. Around the same time a similar idea, the interior penalty (IP) methods, were developed in a series of papers by several authors [55,62,127,349]. These methods allowed for solutions to have discontinuities across element edges, with "penalty" terms used to control jumps in the solution and/or the first derivatives of the solution. Despite extensive analysis for parabolic, elliptic, and hyperbolic problems, IP methods fell out of favor at the time because they seemed to have little computational advantage over continuous Galerkin methods.

In the 1980's and 1990's, DG methods were studied extensively for the solution of hyperbolic conservation laws in a series of papers by Cockburn and Shu [104,106,108–110]. They combined DG spatial discretization with Runge–Kutta time stepping, resulting in the so-called RKDG method. This general approach has been extended for solving hyperbolic systems in complex geometries, including Maxwell's equations, acoustics, elasticity, shallow water equations, free surface waves, plasma physics and gas dynamics [50,58,107,110,119,140,187].

DG and IP methods were revived in the 1990's for elliptic boundary value problems, leading to numerous papers and an alphabet soup of methods, including the nonsymmetric interior penalty Galerkin (NIPG), the symmetric interior penalty Galerkin (SIPG), the incomplete interior penalty Galerkin (IIPG) and local discontinuous Galerkin (LDG) methods. See Arnold et al. [56] for a unified description of these and other DG and IP methods. With improved computational power, $h$ and $p$ adaptive methodologies, and improved linear and nonlinear solvers, DG methods are now competitive with traditional methods [195,215,245,295], and over the past decade there has been an explosion of research in DG methods across a wide spectrum of science and engineering problems [186,305].

Because of their success at solving hyperbolic problems, there has been extensive research in extending the DG methods to advective-dominated advection–diffusion equations, essentially by combining the ideas of the RKDG methods with DG or IP methods for elliptic boundary value problems. As a simple example to demonstrate the basic ideas of the DG method in this case, we consider a one-dimensional advection diffusion equation, which is an instance of Eq. (2)

$$u_t + au_x - cu_{xx} = f, \quad t > 0, \ 0 < x < 1, \tag{65}$$

where subscripts denote partial differentiation, $u(x,0) = u_0(x)$, and for simplicity $u(0,t) = u(1,t) = 0$. The interval $[0,\ 1]$ is divided into elements $\Omega_j = [x_{j-1/2}, x_{j+1/2}]$ of length $h_{e,j}$, with $x_j$ denoting the midpoint of the element, $j = 1,\ldots,n_e$. We consider a test space $V_h$ of functions that are in $H^2$ inside each element, but are not continuous at the interior element interface points $x_{j+1/2}$. Notationally, let

$$w^-(x_{j+1/2}) = \lim_{\epsilon\to 0^-} w(x_{j+1/2}+\epsilon), \tag{66}$$

$$w^+(x_{j+1/2}) = \lim_{\epsilon\to 0^+} w(x_{j+1/2}+\epsilon), \tag{67}$$

$$[\![w(x_{j+1/2})]\!] = w^-(x_{j+1/2}) - w^+(x_{j+1/2}), \text{ and} \tag{68}$$

$$\{w(x_{j+1/2})\} = \frac{1}{2}[w^-(x_{j+1/2}) + w^+(x_{j+1/2})]. \tag{69}$$

Multiplying Eq. (65) by $w \in V_h$, integrating over a single element $\Omega_j$ and integrating by parts we arrive at

$$\int_{\Omega_j} [u_t w - auw_x + cu_x w_x] dx + auw|_{x_{j-1/2}}^{x_{j+1/2}} - cu_x w|_{x_{j-1/2}}^{x_{j+1/2}} = \int_{\Omega_j} f\,v dx. \tag{70}$$

Summing over all $\Omega_j$:

$$\sum_{j=1}^{n_e} \int_{\Omega_j} [u_t w - auw_x + cu_x w_x] dx + \sum_{j=1}^{n_e-1} au[\![w]\!]|_{x_{j+1/2}} - \sum_{j=1}^{n_e-1} cu_x [\![w]\!]|_{x_{j+1/2}}$$
$$= \int_{\Omega_j} fw dx + auw|_{x=0} - auw|_{x=1} - cu_x w|_{x=0} + cu_x w|_{x=1}. \tag{71}$$

Notice that we have not applied any of the boundary conditions to the test space or to the weak formulation at this point. Now, in order to define a DG method, we add several "zero" terms to Eq. (71). These terms are zero because they involve jumps in the true solution, which we assume to be smooth. First, define the "upwind" value of $u$ at $x_{j+1/2}$ as

$$u^{\text{up}} = \begin{cases} u^-, & a > 0 \\ u^+, & a < 0 \end{cases} \tag{72}$$

and define

$$\begin{aligned}\mathcal{A}(u,w) = &-\sum_{j=1}^{n_e} \int_{\Omega_j} [auw_x - cu_x w_x] dx + \sum_{j=1}^{n_e-1} au^{\text{up}} [\![w]\!]|_{x_{j+1/2}} \\ &- \sum_{j=1}^{n_e-1} c\{u_x\}[\![w]\!]|_{x_{j+1/2}} + auw|_{x=1} + cu_x w|_{x=0} - cu_x w|_{x=1} \\ &+ \sum_{j=1}^{n_e-1} \sigma [\![u]\!][\![w]\!]|_{x_{j+1/2}} + \sigma uw|_{x=0} + \sigma uw|_{x=1} \\ &- s_d \left[ \sum_{j=1}^{n_e-1} c[\![u]\!]\{w_x\}|_{x_{j+1/2}} - cuw_x|_{x=0} + cuw_x|_{x=1} \right].\end{aligned} \tag{73}$$

The last six terms in Eq. (73) are zero since $[\![u]\!] = 0$ and $u(0,t) = u(1,t) = 0$. The parameter $s_d$ multiplying the last three terms can be set to 1 (SIPG method), 0 (IIPG method), or $-1$ (NIPG method). The parameter $\sigma$ is called the penalty parameter and must be chosen to be sufficiently large in the SIPG or IIPG methods for stability, and must be at least positive in the NIPG method. Note that the first choice results in a symmetric operator $\mathcal{A}$ when $a = 0$ and the latter two result in a nonsymmetric operator. Thus Eq. (71) can be rewritten succinctly as

$$\sum_{j=1}^{n_e} \int_{\Omega_j} u_t w dx + \mathcal{A}(u,w) = \sum_{j=1}^{n_e} \int_{\Omega_j} fw dx. \tag{74}$$

The LDG formulation follows a different path and rewrites Eq. (71) in a mixed formulation by introducing an auxiliary variable

$$\kappa_d = -cu_x. \tag{75}$$

Next define an approximating space

$$W_h = \{w : w \in \mathcal{P}^k(\Omega_j),\ j = 1,\ldots,n_e\}, \tag{76}$$

which is a finite dimensional subspace of $V_h$, where $\mathcal{P}^k$ denotes the set of all polynomials of degree less than or equal to $k$, $k \geqslant 1$. Finally, the continuous time DG formulation of Eq. (74) is to find $u_h \in W_h$ such that

$$\sum_{j=1}^{n_e} \int_{\Omega_j} \frac{\partial u_h}{\partial t} w_h dx + \mathcal{A}(u_h, w_h) = \sum_{j=1}^{n_e} \int_{\Omega_j} fw_h dx, \quad \forall w_h \in W_h. \tag{77}$$

This equation can now be integrated in time using any number of temporal discretization methods.

The applications of DG methods to systems of advection–diffusion-reaction equations, shallow water, and Navier–Stokes flow equations, are very active areas of research. These problems are typically advective-dominated, can give rise to shocks and/or sharp fronts in the solution, require unstructured grids to resolve complex geometric and/or solution features, and are thus well-suited for solution by DG methods. While there has been substantial research at applying DG methods to subsurface flow problems, the methods have not generally been adopted in this community, primarily because of their expense and the traditional use of low-order finite difference methods. The transfer of DG technology into production or operational codes is still years away, primarily because of questions of cost and efficiency. However, as computing architectures continue to evolve, requiring new paradigms in algorithms and solvers, and with the requirements of multiphysics and multiscale modeling, DG methods may well prove to be a foundational method for solving PDEs on the computing platforms of the future [117,237].

### 4.2. Multiscale methods

Many problems in water resources involve multiple length scales. Several examples of such problems were discussed in Section 2. The direct numerical solution of multiscale problems is difficult, and frequently impossible, even with modern high-performance computers because of both computational effort and memory limitations. However, from an application perspective, it is often sufficient to predict the properties of a multiscale system on a scale that is much coarser than the small length scale that influences the behavior. Therefore, it is desirable to develop a method that captures the small-scale effect on the large scales, but does not require resolving all the small-scale features.

In response to this need, multiscale methods have received considerable attention over the last several years. Theory has been advanced to support multiscale methods and analyze their behavior, and a growing body of comparisons among different candidate multiscale methods has been reported. It is clear that multiscale methods, broadly speaking, will be an important area of both research and application in the applied mathematics and water resources fields for the foreseeable future.

We summarize some aspects of multiscale methods in the sections that follow: general concepts of the multiscale finite element method (MsFEM) [136,193], approaches to improve multiscale basis functions, the multiscale finite volume method (MsFVM) [210], generalization of MsFEM to nonlinear problems, coarse grid choices, systematic enrichment of multiscale spaces, multiscale methods for stochastic problems, mixed MsFEM [45,100], and an application to multiphase flow and transport processes. The broad scope of this treatment is indicative of both the body of work that has been performed and the prominence of multiscale methods in modern science.

### 4.2.1. General concepts of MsFEM

MsFEM represents small-scale behavior through the calculation of localized basis functions that depend upon small-scale aspects of the PDAE model. These basis functions contain essential multiscale information embedded in the solution and are coupled through a global formulation to provide a faithful approximation of the solution. The MsFEM originated from [60,61] where oscillatory basis functions were constructed for elliptic equations with special heterogeneous coefficients. There are now several approaches to capture the effect of small scales on the large-scale solution. In this section we focus on a simple MsFEM to introduce some basic notions, which are covered in more detail in the literature [e.g. 136,193].

Consider a Galerkin finite element method formulation for a linear elliptic PDE of the form

$$Lp = f \quad \text{in } \Omega, \tag{78}$$

where $\Omega$ is a domain in $\mathbb{R}^d (d = 2,3)$, $L$ is a differential operator, $Lp := -\nabla \cdot (\mathbf{k} \cdot \nabla p)$, $\mathbf{k} = k_{ij}(\mathbf{x})$ is a heterogeneous permeability tensor with multiple scales and is assumed to be symmetric and satisfies $\alpha|\xi|^2 \leqslant k_{ij}\xi_i\xi_j \leqslant \beta|\xi|^2$, for all $\xi \in \mathbb{R}^d$ with $0 < \alpha < \beta$. We note that $L$ can be a different operator, e.g. representing other water resources models.

MsFEM consists of two major ingredients: (1) multiscale basis functions, and (2) a global formulation that couples the multiscale basis functions. Important multiscale features of the solution are incorporated into these localized basis functions that can contain information about the scales that are both smaller and larger than the local scale defined by the localized basis functions. A global formulation couples these localized basis functions to approximate the solution.

The basis function construction for the Galerkin MsFEM is straightforward for the common case of resolving sub-scale detail. Let $\mathcal{T}_h$ be a usual partition of $\Omega$ into simplices (e.g. triangles, quadrilaterals). We call this partition the coarse grid and assume that the coarse grid is resolved with a fine grid, which is depicted in Fig. 1. Let $\mathbf{x}_i$ be the interior coarse nodes of the mesh corresponding to $\mathcal{T}_h$ and $\phi_i^0$ be the nodal basis of a standard finite element space $W_h$. To formulate the equation describing multiscale basis functions, we denote by $S_i = \text{support}(\phi_i^0)$. Next, we define $\phi_i$ with support in $S_i$ as follows

$$L\phi_i = 0 \ \text{in } K, \qquad \phi_i = \phi_i^0 \ \text{on } \partial K, \quad \forall K \in \mathcal{T}_h. \tag{79}$$

Note that $\partial K$ represents all edges of the coarse grid block (see Fig. 1). Solution of Eq. (79) for each basis function $\phi_i$ yields the finite element space spanned by $\phi_i$, which is denoted

$$\mathcal{P}_h = \text{span}\{\phi_i\}. \tag{80}$$

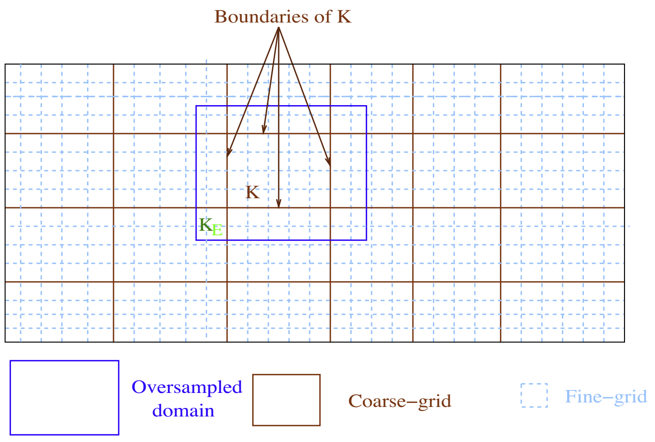

**Fig. 1.** Schematic description of an oversampled region.

The solution of Eq. (79) depends upon the operator $L$, the structure of the coefficients that appear in $L$, and the spatial dimensionality of the problem. For the elliptic operator defined above, an analytical solution may be possible and is trivial in one spatial dimension [e.g. 136]. Note that multiscale basis functions are constructed off-line and can be used to solve the global problem for any source terms and right-hand sides. In particular, for multiphase flow simulations, multiscale basis functions are constructed based on single-phase flow information and this reduces the computational cost of the computations.

Our main assumption is that the basis functions satisfy the leading order homogeneous equations when the right hand side $f$ of Eq. (78) is a smooth function. This assumption allows the MsFEM formulation to represent systems that have a separation of length scales. If the scale of variations are small compared to the length scale of the simplices, one can choose $K$ to have a length scale smaller than the coarse grid. The exact shape of multiscale basis functions will depend upon the sub-scale field being resolved.

The global formulation of MsFEM is analogous to the approach taken in standard Galerkin finite element methods with conforming basis functions ($\mathcal{P}_h \subset H_0^1(\Omega)$). For MsFEM the global formulation requires finding $p_h \in \mathcal{P}_h$ such that

$$\int_\Omega \nabla v_h \cdot \mathbf{k} \cdot \nabla p_h \mathrm{d}\mathbf{x} = \int_\Omega f v_h \mathrm{d}\mathbf{x} \quad \forall v_h \in \mathcal{P}_h. \tag{81}$$

Test functions can be chosen from $W_h$ (instead of $\mathcal{P}_h$). This will lead to the Petrov–Galerkin version of multiscale finite element method as introduced in [192]

$$\int_\Omega \nabla v_h \cdot \mathbf{k} \cdot \nabla p_h \mathrm{d}\mathbf{x} = \int_\Omega f v_h \mathrm{d}\mathbf{x} \quad \forall v_h \in W_h. \tag{82}$$

In both Eqs. (81) and (82), the test functions correspond to the coarse scale $\mathcal{T}_h$. The computation of the global coefficient matrix corresponding to either Eq. (81) or Eq. (82) requires an appropriate integration, which may be analytically performed in simple cases or require an appropriate quadrature scheme for more complicated cases. Note the computation of the global coefficient matrix can be performed off-line thus there is no need to do integration over the fine mesh if multiscale basis functions do not change.

If the local computational domain is chosen to be smaller than the coarse grid block, then one can use an approximation of $p_h$ in a representative volume element (RVE) to represent the left-hand side of Eq. (82). Since the RVE within each coarse grid block represents the small-scale features of the media, one can approximate $\mathbf{k} \cdot \nabla p_h$ by a periodic function. In this case, there is no need to compute the integral over the entire coarse grid block $K$ and one can approximate these integrals via the integrals over the RVE by evaluating

$$\int_\Omega \nabla v_h \cdot \mathbf{k} \cdot \nabla p_h \mathrm{d}\mathbf{x} \approx \sum_K \frac{|K|}{|K|_{loc}} \int_{K_{loc}} \nabla v_h \cdot \mathbf{k} \cdot \nabla p_h \mathrm{d}\mathbf{x}, \tag{83}$$

where $K_{loc}$ refers to a local computational region (RVE). One can approximate the right hand side of Eq. (82) similarly yielding

$$\sum_K \frac{|K|}{|K|_{loc}} \int_{K_{loc}} \nabla v_h \cdot \mathbf{k} \cdot \nabla p_h \mathrm{d}\mathbf{x} = \sum_K \frac{|K|}{|K|_{loc}} \int_{K_{loc}} f v_h \mathrm{d}\mathbf{x} \quad \forall v_h \in W_h. \tag{84}$$

The choice of boundary conditions in defining the multiscale basis functions plays an important role in approximating the solution. The boundary condition imposed for the multiscale basis function needs to reflect the multiscale oscillations of the solution on the boundaries of coarse regions. If we choose a linear or any other "artificial" boundary condition for the basis function, a mismatch is created between the exact solution and the multiscale finite element approximation along the boundaries of coarse-grid blocks that can result in large errors. We will briefly discuss this issue and approaches to reduce the subgrid capturing errors.

### 4.2.2. Improving multiscale basis functions

The boundary conditions for the basis functions play an important role in the accuracy of MsFEMs. Without proper oscillatory boundary conditions, MsFEM can have large errors. These errors are mainly due to the resonance between the coarse grid size and the characteristic length scale of the problem. For the case where $\mathbf{k}(\mathbf{x})$ is a periodic function varying over $\varepsilon$ scale ($\mathbf{k}(\mathbf{x}) = \mathbf{k}(\mathbf{x}/\varepsilon)$), the convergence rate of MsFEM is $\varepsilon/h$. This error is large when $h \approx \varepsilon$. By a judicious choice of boundary conditions, these errors can be reduced.

To reduce the influence of linear boundary conditions, Hou and Wu [194] proposed using larger sampling regions in the construction of the multiscale trial space. Specifically, let $\psi_j$ be the basis functions (or local solutions) satisfying the homogeneous elliptic equation in the larger domain $K_E \supset K$ (see Fig. 1). Then, the actual basis $\phi_i$ are formed by using the linear combination of $\psi_j$ given by

$$\phi_i = \sum_{j=1}^{d} c_{ij} \psi_j. \tag{85}$$

The coefficients $c_{ij}$ are calculated by the condition $\phi_i(\mathbf{x}_j) = \delta_{ij}$, where $\mathbf{x}_j$ are the locations of the coarse-grid nodal points, and $\delta_{ij}$ is the Kronecker delta function. Numerical experiments have demonstrated that the oversampling technique reduces the numerical error; however, the oversampling technique results in a *non-conforming* MsFEM method [e.g. 137].

In a number of approaches [e.g. 135,288], a limited number of global solutions are computed and used to construct multiscale basis functions. This improves the accuracy of the method when there are global features that need to be represented on a coarse grid; refer to [136] for further discussion.

### 4.2.3. Comparison to other multiscale methods

MsFEM shares similarities with other multiscale methods. In this section, we briefly discuss the similarity to flow-based upscaling techniques and refer to [136] for comparisons to other methods such as variational multiscale methods and heterogeneous multiscale methods.

One of the early approaches is the flow-based upscaling technique in the context of porous media [e.g. 128,356]. The main idea of flow-based upscaling techniques is to form a coarse-scale equation and pre-compute the effective coefficients based on offline simulations similar to the construction of multiscale basis functions. For linear elliptic equations, the coarse-scale equation has the same form but with modified coefficients. The effective coefficients are computed using local solutions in a representative volume (i.e., the upscaling is based on the flow equation). We note that there are many other upscaling techniques where various averaging techniques are used to compute the upscaled permeability field. Here, we focus on the comparisons to flow-based upscaling techniques that are rigorous from the point of view of averaging small scales.

We consider local solutions described by

$$\nabla \cdot (\mathbf{k} \cdot \nabla \phi_e) = 0 \text{ in } K \tag{86}$$

with $\phi_e(\mathbf{x}) = \mathbf{x} \cdot \mathbf{e}$ on $\partial K$, where $\mathbf{e}$ is a unit vector. Here $K$ denotes a coarse grid block, as before. The effective coefficients are computed in each $K$ as

$$\mathbf{k}^* \cdot \mathbf{e} = \int_K \mathbf{k} \cdot \nabla \phi_e \mathrm{d}\mathbf{x}. \tag{87}$$

Note that $\mathbf{k}^*$ is a symmetric matrix provided $\mathbf{k}$ is symmetric. One can employ various boundary conditions that include periodic boundary conditions and oversampling techniques as well as the use of limited global information. We refer to [128,356] for the discussion on the use of various boundary conditions. Once the effective coefficients are calculated, the global coarse-scale equation

$$\nabla \cdot (\mathbf{k}^* \cdot \nabla p^*) = f \tag{88}$$

is solved over the entire region.

To show the similarity to MsFEM, the discretization of Eq. (88) using a Petrov–Galerkin finite element method is

$$\int_\Omega \nabla v_h \cdot \mathbf{k}^* \cdot \nabla p_h^* \mathrm{d}\mathbf{x} = \int_\Omega f v_h \mathrm{d}\mathbf{x}, \quad \forall v_h \in W_h, \tag{89}$$

where $p_h^* \in W_h$. Next, we substitute Eq. (87) into Eq. (89). If we assume that the coarse grid consists of linear triangle simplices, thus, $\nabla p_h^*$ is constant within each coarse grid block, then it follows that

$$\mathbf{k}^* \cdot \nabla p_h^* = \int_K \mathbf{k} \cdot \nabla \phi_{\nabla p_h^*} \mathrm{d}\mathbf{x}, \tag{90}$$

where $\phi_{\nabla p_h^*}$ solves the local problem with $\nabla p_h^* \cdot \mathbf{x}$ Dirichlet boundary conditions. In summary, we have the following formulation for the coarse-scale equation

$$\sum_{K \in \mathcal{T}_h} \int_K \nabla v_h \cdot \mathbf{k} \cdot \nabla \phi_{\nabla p_h^*} \mathrm{d}\mathbf{x} = \int_\Omega f v_h \mathrm{d}\mathbf{x}, \quad \forall v_h \in W_h. \tag{91}$$

Since $\phi_{\nabla p_h^*}$ can be written as a linear combination of multiscale finite element basis functions $\phi_i$, this shows that MsFEM can be derived from flow-based upscaling methods. However, MsFEM differs from traditional upscaling methods, since the local information can be recovered adaptively. Moreover, in MsFEM methods, one can enforce limited global information and enrich multiscale spaces that are not, in general, possible, for flow-based upscaling techniques.

### 4.2.4. Multiscale finite volume method (MsFVM)

Mass conservative schemes play a central role in subsurface applications. Next, we consider methods that can provide a mass conservative approximation for the flux defined by $\mathbf{k} \cdot \nabla p$. Above, we presented a general concept of MsFEM that can be used to develop mass-conservative schemes. In particular, the global formulation can be changed and we can use various global formulations based on finite volume, mixed finite element, DG methods, etc. Some of these approaches have been studied in the literature (see [136]).

One of these methods within a finite volume context was first proposed in [210]. The main idea of this method is to use a mass conservative finite volume global coupling for multiscale basis functions. In the latter approach, MsFVM is used as a global coupling for multiscale basis functions. In particular, as a test function, piecewise constant function supported on a target coarse-grid block are used to couple multiscale basis functions that are defined on a dual coarse-grid block. Denoting the control volume corresponding to a coarse node $\mathbf{x}_i$ by $V_i$, we consider the coarse space spanned by multiscale finite element basis functions defined in $V_i$ ($V_i$ replaces $K$ in (79)). The global formulation of MsFVM is defined as

$$\int_{\partial K} \mathbf{n} \cdot \mathbf{k} \cdot \nabla p_h \mathrm{d}l = \int_K f \mathrm{d}\mathbf{x}, \tag{92}$$

for every control volume $K \subset \Omega$, where $\mathbf{n}$ denotes the outward normal vector on the boundary. The resulting multiscale method differs from MsFEM since it employs the finite volume element method as a global solver. Note that the coarse-scale velocity field obtained using MsFVM is conservative in $\mathcal{T}_h$.

One can modify basis functions and take into account oversampling or limited global information. This has been tested for two-phase flow and transport where the single-phase flow solution is used to construct multiscale basis functions (see [135] for the results when control volumes are chosen in the global formulation (92)). The construction guarantees that the basis functions span the single-phase flow solution. We have observed substantial improvement in the simulation results by incorporating limited global information in the construction of multiscale basis functions.

### 4.2.5. Generalization of MsFEM to nonlinear problems

MsFEMs can be generalized to nonlinear problems without involving a linearization step. Next, we briefly discuss it in the context of an abstract nonlinear equation

$$Lp = f, \tag{93}$$

where $L : X \to Y$ is a nonlinear differential operator for which $L$ and $f$ are known and the solution involves finding $p$. Examples of $L$ can include Richards' equation, coupled fluid–structure interaction, nonlinear monotone operators that arise in Forchheimer flow and other nonlinear systems of coupled equations, in general. Let $W_h$ be a finite dimensional space with an approximation property [358], and $h$ is the coarse-grid mesh size. For the nonlinear MsFEM, multiscale basis functions are replaced by *multiscale maps* defined as $E^{ms} : W_h \to V_h$, which are defined for each element $v_h \in W_h$, $v_{r,h} = E^{ms} v_h$ where

$$L^{map} v_{r,h} = 0 \text{ in } K, \tag{94}$$

where $L^{map}$ can be different from $L$ and captures the effects of the small scales. Moreover, domains different from the target coarse block $K$ can be used to compute local solutions. Boundary conditions are needed to solve Eq. (94) where we refer to earlier discussions on concepts of selecting boundary conditions. We seek a solution of Eq. (93) in $V_h$ as follows. Find $p_h \in W_h$ (consequently $p_{r,h} \in V^h$) such that

$$\langle L^{global} p_{r,h}, v_h \rangle = \langle f, v_h \rangle, \quad \forall v_h \in W_h, \tag{95}$$

where $\langle u, v \rangle$ denotes the duality between $X$ and $Y$, and $L^{global}$ can be different from $L$. The correct choices of $L^{map}$ and $L^{global}$ are needed for the accuracy and convergence of MsFEM. For linear elliptic equations, $L^{map}$ is a linear map, and thus, $V_h$ is a linear space spanned by $E^{ms}\phi$, where $\phi \in W_h$.

We note that the general concept of MsFEM is used for nonlinear problems, such as Richards' equation, coupled pore-scale and Darcy-scale models for deformable porous media, and pseudo-monotone operators (see [136] for further details). In all these cases, various $L^{map}$ and $L^{global}$ are constructed and convergence of the methods are studied. In coupled pore-scale and Darcy-scale models, $L^{map}$ consists of local fluid–structure interaction problem given pressure and displacements at the coarse nodes, while $L^{global}$ consists of coupled Darcy and elasticity equations. For steady-state Richards' equation, $Lp = \nabla \cdot (\mathbf{k}(\mathbf{x}, p) \cdot \nabla p)$, $L^{map}$ is described by $L^{map}\phi = \nabla \cdot (\mathbf{k}(\mathbf{x}, \eta^\phi) \cdot \nabla \phi) = 0$, where $\eta^\phi = \frac{1}{|K|}\int_K \phi \mathrm{d}\mathbf{x}$, while $L^{global}$ is obtained by taking the test functions to be piecewise linear functions as in Petrov–Galerkin formulation of MsFEM. Note that when there is a separation between nonlinearities and heterogeneities, $\mathbf{k}(\mathbf{x}, \eta^\phi) = \mathbf{k}(\mathbf{x}) k_r(\eta^\phi)$, then one can use the same solution $\phi$ of $L^{map}\phi = 0$ for all $\eta^\phi$ provided $k_r(\eta^\phi)$ is a smooth function. This allows reuse of the multiscale basis functions throughout the simulations. Otherwise, multiscale basis functions need to be re-computed if the changes in heterogeneities due to $\eta^\phi$ is large. As for the global formulation, we seek the coarse-grid solution, $p_h$, such that

$$\int_\Omega \nabla v_h \cdot \mathbf{k}(\mathbf{x}, \eta^{p_h}) \cdot \nabla p_h \mathrm{d}\mathbf{x} = \int_\Omega f v_h \mathrm{d}\mathbf{x} \quad \forall v_h \in W_h. \tag{96}$$

One can also use finite volume framework to obtain mass-conservative solution.

### 4.2.6. Coarse grid choice

MsFEM can be applied not only to regularly spaced grids, but to unstructured grids. The only gridding requirement is that every coarse grid consists of a connected union of fine-grid blocks. In [44], the authors developed a gridding technique that used the single-phase flow information to construct a coarse grid. The coarse grid was chosen such that it minimized the global single-phase velocity vector field. This automatic coarse-grid generator allows one to use an optimal coarse grid for simulation purposes. Finally, the fine-grid blocks in neighboring coarse grid blocks do not need to match along the interface.

### 4.2.7. Systematic enrichment of multiscale spaces

One can systematically enrich coarse spaces by adding new basis functions. The construction of these basis functions depends on initial multiscale basis functions [134]. With a careful choice of initial multiscale basis functions, a small dimensional coarse space can be achieved. The enrichment of the coarse spaces uses local spectral problems with the weight function depending on initial multiscale basis functions. We briefly review the results of [134]. Consider the eigenvalue problem

$$\nabla \cdot (\mathbf{k} \cdot \nabla \psi_l^{\omega_i}) = \lambda_l^{\omega_i} \tilde{k} \psi_l^{\omega_i}, \tag{97}$$

where $\lambda_l^{\omega_i}$ (or simply $\lambda_l^i$) and $\psi_l^{\omega_i}$ (or simply $\psi_l^i$) are eigenvalues and eigenvectors in $\omega_i$ and $\tilde{k}$ is defined by

$$\tilde{k} = \sum_j \nabla \phi_j \cdot \mathbf{k} \cdot \nabla \phi_j, \tag{98}$$

where $\phi_j$ are initial multiscale basis functions as discussed earlier and a sum is taken over all coarse nodes. The eigenvalue problem formulated above is solved with zero Neumann boundary condition, and it is solved in a discrete setting. Assume eigenvalues are given by

$$0 = \lambda_1^{\omega_i} \leqslant \lambda_2^{\omega_i} \leqslant \ldots. \tag{99}$$

Basis functions are computed by selecting a number of eigenvalues (starting with small ones) and multiplying the corresponding eigenvectors by $\phi_i$. Thus, the multiscale space is defined for each $i$ as the span of $\phi_i \psi_l^{\omega_i}$, $l = 1, \ldots, L_i$, where $L_i$ is the number of selected eigenvectors.

The dimension of the coarse space depends on the choice of $\tilde{k}$ and thus it is important to have a good choice of $\tilde{k}$ when solving local eigenvalue problem. The essential ingredient in designing $\tilde{k}$ is to guarantee that there are fewer small, asymptotically vanishing (when contrast increases) eigenvalues of Eq. (97). With an initial choice of multiscale basis functions that contain many small-scale localizable features of the solution, one can reduce the dimension of the coarse space as demonstrated in [134]. In particular, we note that the eigenvectors corresponding to small, asymptotically vanishing (when contrast increases) eigenvalues represent the local features of the solution. These features typically are not captured by the initial multiscale basis functions. This gives a natural way to complement the initial coarse space. The initial multiscale spaces are important because a poor choice of initial basis functions can result in a large dimensional coarse space. Thus, the use of advanced multiscale techniques in constructing initial basis helps reduce the dimension of the coarse space that is needed to achieve contrast-independent two-level domain decomposition preconditioners and more accurate coarse-grid solutions. In [134], the authors showed that under some conditions the convergence rate is inversely related to the smallest eigenvalue whose eigenvector is not included in the coarse space.

We note that with an appropriate choice of coarse space one can achieve optimal preconditioners that converge independent of physical parameters such as small scales and high contrast. In particular, choosing coarse spaces as above is needed for obtaining robust preconditioners. We refer to [160] where advantages of using multiscale basis functions in domain decomposition methods are demonstrated. The multiscale basis functions are further improved to take into account high-contrast coefficients in the work [103].

The success of MsFEMs depends whether the space of local snapshots can be represented with fewer basis functions. This can be successfully done for many elliptic and parabolic problems. However, how much model reduction can achieved for other processes, e.g. convection or reaction dominated processes, needs further study.

#### *4.2.8. Multiscale methods for stochastic problems and parameter-dependent problems*

In many subsurface applications, the media properties contain uncertainties. After parameterizing these uncertainties, one faces a challenging task of solving many forward problems where each of these forward problems contains multiple scales. Because the multiscale basis construction can require elaborate procedures to identify reduced-order local models, we can consider constructing multiscale basis functions for an ensemble of realizations. In [43,133], MsFEMs have been generalized and applied to stochastic and parameter-dependent problems. In these problems, multiscale basis functions are constructed in the off-line stage by selecting a number of random realizations of the permeability fields. These local multiscale basis functions are orthogonalized and used in the online stage to solve the flow problem for a randomly selected realization of the permeability field. In particular, in [133], new multiscale basis functions are computed in the online stage by solving a small local problem that consists of projecting the local solution onto an existing set of multiscale basis functions. In [43], the authors use limited global information in multiscale basis construction. In this case, all local basis functions are kept in the online stage to compute the solution of flow problems. Both approaches show that via the off-line computations of multiscale basis functions, one can reduce the computational cost at the online stage involving the construction of multiscale basis functions for a wide range of stochastic and parameter-dependent flow problems.

#### *4.2.9. Mixed multiscale finite element method*

MsFVEM provides a mass conservative velocity field (defined as $\mathbf{v} = -\mathbf{k}\cdot\nabla p$) on the coarse grid. However, the fine-scale velocity field reconstructed using multiscale basis functions is no longer conservative on the fine grid. For multiphase flow and transport simulations, the conservative fine-scale velocity field is often needed. A mixed MsFEM can be formulated to provide such a conservative fine-scale velocity field [42,52,100].

To present the mixed MsFEM (following [100]), we re-write the elliptic equation in the form

$$\begin{aligned} &\mathbf{k}^{-1}\cdot\mathbf{v} + \nabla p = 0 \quad \text{in } \Omega \\ &\nabla\cdot\mathbf{v} = f \quad \text{in } \Omega \end{aligned} \tag{100}$$

with non-homogeneous Neumann boundary conditions $\mathbf{v}\cdot\mathbf{n} = 0$ on $\partial\Omega$. In the mixed multiscale finite element method, the basis functions for the velocity field, $\mathbf{v} = -\mathbf{k}\cdot\nabla p$, are needed. As in the case of MsFEM, one can use known mixed finite element spaces to construct these basis functions. We consider multiscale basis functions corresponding to the lowest order Raviart–Thomas elements [52,100]. The basis functions for the velocity in each coarse block $K$ are given by

$$\begin{aligned} &\nabla\cdot(\mathbf{k}\cdot\nabla w_i^K) = \frac{1}{|K|} \quad \text{in } K \\ &\mathbf{n}\cdot\mathbf{k}\cdot\nabla w_i^K = \begin{cases} g_i^K & \text{on } e_i^K \\ 0 & \text{else,} \end{cases} \end{aligned} \tag{101}$$

where $g_i^K = \frac{1}{|e_i^K|}$ and $e_i^K$ are the edges of $K$. We define the finite dimensional space for velocity by

$$V_h = \text{span}\{\Psi_i^K\}, \tag{102}$$

where $\Psi_i^K = \mathbf{k}\cdot\nabla w_i^K$. The basis functions for the pressure are piecewise constant functions over each $K$. We denote the span of these basis functions by $Q_h$. The basis functions for the velocity field, $\Psi_i^K$, are conservative on both the fine and coarse grids if the local problems are solved using a conservative scheme. The mixed finite element framework couples the velocity and pressure basis functions. This provides an approximation of the global solution for both pressure and velocity.

Multiscale basis functions can be constructed using information from the global fields or fields defined in larger regions, $\mathbf{v}_1,\ldots,\mathbf{v}_N$, which can be obtained from the solutions of single-phase flow equations. The basis function for velocity can be constructed as

$$\begin{aligned} &\nabla\cdot\left(\mathbf{k}\cdot\nabla\phi_{ij}^K\right) = \frac{1}{|K|} \quad \text{in } K \\ &\mathbf{n}_{e_l}^K\cdot\mathbf{k}\cdot\nabla\phi_{ij}^K = \delta_{jl}\frac{\mathbf{v}_i\cdot\mathbf{n}_{e_l}^K}{\int_{e_l}\mathbf{v}_i\cdot\mathbf{n}_{e_l}ds} \quad \text{on } \partial K \\ &\int_K \phi_{ij}^K dx = 0, \end{aligned} \tag{103}$$

where $i = 1,\ldots,N$ and $j = 1,2,3$ (assuming $K$ is a triangle), $\delta_{jl}$ is the Kronecker delta function, and $e_l$ denotes an edge of the triangle. One can see that for each edge, there are $N$ basis functions with the normal flux equal to the normalized $v_i\cdot\mathbf{n}$. The normal flux along other edges is zero. Because of the latter, a source term is required for Eq. (103). We define $\boldsymbol{\psi}_{ij}^K = \mathbf{k}\cdot\nabla\phi_{i,j}^K$ and the finite dimensional space spanned by these basis functions by

$$V_h = \text{span}\{\psi_{ij}^K\}. \tag{104}$$

We denote by $V_h^0$ the span of $\psi_{ij}^K$ that satisfies homogeneous Neumann boundary conditions. Let $Q_h$ be piecewise constant basis functions that are used to approximate $p$. Note that for each edge, we have $N$ basis functions and we assume that $\mathbf{v}_1,\ldots,\mathbf{v}_N$ are linearly independent and $|\int_{e_l}\mathbf{v}_i\cdot\mathbf{n}ds|$ is bounded below to avoid the possibility of $\int_{e_l}\mathbf{v}_i\cdot\mathbf{n}ds = 0$.

The mixed formulation is to find $\{\mathbf{v},p\}$ such that $\mathbf{v}\cdot\mathbf{n} = g$ on $\partial\Omega$ and

$$\begin{aligned} &\int_\Omega \mathbf{w}\cdot(\lambda\mathbf{k})^{-1}\cdot\mathbf{v}d\mathbf{x} + \int_\Omega \nabla\cdot\mathbf{w}pd\mathbf{x} = 0, \quad \forall\mathbf{w}\in H_0(div,\Omega) \\ &-\int_\Omega \nabla\cdot\mathbf{v}qd\mathbf{x} = \int_\Omega fqd\mathbf{x}, \quad \forall q\in L^2(\Omega)/R, \end{aligned} \tag{105}$$

where $L^2(\Omega)/R$ is $L^2$ space factorized by constants. One can easily generalize the method to handle non-homogeneous boundary conditions.

To formulate the mixed multiscale finite element method, we use conforming basis functions, e.g. basis functions associated with the lowest order Raviart–Thomas mixed multiscale finite element. The global formulation is to find $\{\mathbf{v}_h, p_h\} \in V_h \times Q_h$ such that $\mathbf{v}_h \cdot \mathbf{n} = 0$ on $\partial\Omega$. We can form

$$\begin{aligned} &\int_\Omega \mathbf{w}_h \cdot \mathbf{k}^{-1} \cdot \mathbf{v}_h d\mathbf{x} - \int_\Omega \nabla \cdot (\mathbf{w}_h) p_h d\mathbf{x} = 0, \quad \forall \mathbf{w}_h \in V_h^0 \\ &\int_\Omega \nabla \cdot \mathbf{v}_h q_h d\mathbf{x} = \int_\Omega f q_h d\mathbf{x}, \quad \forall q_h \in Q_h, \end{aligned} \tag{106}$$

where $V_h^0$ is a subspace of $V_h$ with elements that are homogeneous on the boundary. The above formulation is the mixed multiscale finite element method introduced in [100]. This method was modified later by Aarnes in [42] when applied to porous media flow simulations.

#### 4.2.10. An application to multiphase flow and transport processes

MsFEM and their modifications are used in multiphase flow and transport simulations through heterogeneous porous media in the presence of gravity, compressibility, and so on. The main idea of these approaches is to compute multiscale basis functions for flow equations and re-use or adaptively re-compute multiscale basis functions in some selected regions. The methods are tested for challenging heterogeneous cases and extended to transport equations. We refer to a vast literature [e.g. 42,45,136,179,180,209,210,251, 263–265] and the references therein.

The discussion about the performance, accuracy, and speed-up can be found in the literature [e.g. 136]. For multiphase flow and transport simulations, the main computational speed-up is due to off-line computations of multiscale basis functions. Multiscale basis functions can be re-used at every step of two-phase flow simulations and thus the pressure equation is solved on the coarse grid. This can provide a substantial speed-up.

## 4.3. Krylov deferred correction methods in geochemical applications

For temporal discretizations, the most commonly used schemes include linear multistep methods (e.g. backward difference formulas) and Runge–Kutta methods; existing packages include DASPK [85,266] and RADAU5 [177,178]. In recent years, spectral and pseudospectral formulations have been "re-searched" for initial differential equation problems requiring high-accuracy results [170,176], and they are often solved accurately and efficiently using the Picard type integral equation formulations, Gaussian quadrature, and deferred correction accelerations [130,197]. This evolving class of methods has potential, but unrealized, application to water resources problems.

### 4.3.1. Krylov deferred correction methods

Consider the pseudospectral Legendre–Gauss collocation method for integrating a simple first-order ODE system in one time step from 0 to $\Delta t$,

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0. \tag{107}$$

Let $c_1, \ldots, c_s$ be the linearly scaled Gaussian nodes on $[0, 1]$. The collocation polynomials $\mathbf{u}(t)$ of degree $s$ over $[0, \Delta t]$ satisfy

$$\mathbf{u}(0) = \mathbf{y}_0, \qquad \mathbf{u}'(c_i\Delta t) = \mathbf{f}[c_i\Delta t, \mathbf{u}(c_i\Delta t)], \quad i = 1, \ldots, s, \tag{108}$$

and the numerical solution at $t = \Delta t$ is given by $\mathbf{u}(\Delta t)$. In [168,355], Eq. (108) is shown to be equivalent to the $s$-stage Runge–Kutta method, and is therefore often referred to as the Gauss Runge–Kutta (GRK) method. Numerical properties of the GRK method have been well-studied [e.g. 170,176], and it is well-known that the GRK method with $s$ nodes for the ODE system given by Eq. (107) is order $2s$ (super-convergent), $A$-stable, $B$-stable, symplectic (structure-preserving), symmetric (time reversible), and the error decays exponentially when $s$ increases for fixed time step size.

Despite these excellent properties, notice that the unknowns in the GRK formulation given by Eq. (108) are coupled and the solution at the current time depends on future times, while in the original system given by Eq. (107), the solution only depends on data in the past. For a system of $N$ nonlinear equations, direct Gauss elimination of the linearized equations at each Newton iteration requires prohibitive $O(N^3s^3)$ work for each time marching step. Therefore, a higher order GRK formulation is seldom used due to efficiency considerations, the excellent mathematical properties notwithstanding.

In 2000, Dutt et al. introduced an iterative *spectral deferred correction* (SDC) method [130] for the efficient solution of Eq. (108), or the corresponding discretized Picard integral equation

$$\mathbf{y}(t) = \mathbf{y}(0) + \int_0^t \mathbf{f}[\tau, \mathbf{y}(\tau)] d\tau. \tag{109}$$

In the SDC method, using an existing low-order scheme (e.g. Euler's method), a provisional solution is first obtained at the Gaussian nodes in one time step $[0,\ \Delta t]$ and the corresponding interpolating polynomial, which is constructed using the numerically stable least squares based Legendre polynomials and Gaussian quadrature, is denoted as $\tilde{\mathbf{u}}(t)$. Second, as the low order $\tilde{\mathbf{u}}(t)$ does not satisfy the higher order discretization in Eq. (108), an equation for the error $\boldsymbol{\delta}(t) = \mathbf{y}(t) - \tilde{\mathbf{u}}(t)$ is defined by plugging $\tilde{\mathbf{u}}(t) + \boldsymbol{\delta}(t)$ in place of $\mathbf{y}(t)$ in Eq. (107). Third, the low-order scheme is applied to the error equation to derive a lower order approximation $\tilde{\boldsymbol{\delta}}(t)$ of the error $\boldsymbol{\delta}(t)$, and $\tilde{\boldsymbol{\delta}}$ is then added to the provisional solution $\tilde{\mathbf{y}}$ in order to form a better approximation. This iteration continues for a prescribed number of times or until a prescribed error tolerance is achieved. Clearly, if the iterations converge, the solution will satisfy the collocation formulation in Eq. (108).

Unfortunately, for general DAEs of higher-index, it has been demonstrated numerically that the analogous SDC iteration procedure becomes divergent for many systems [196]. Analysis in [197] shows that for linear problems, the SDC is equivalent to a preconditioned Neumann series expansion, where the preconditioner is the low-order deferred correction procedure. When there exist a few "bad" eigenvalues in the series expansion, one observes order reduction for stiff problems or divergence for certain DAEs. To improve the convergence, in [197], the Krylov deferred correction (KDC) method was introduced. Instead of simply adding the low order $\tilde{\boldsymbol{\delta}}(t)$ to $\tilde{\mathbf{y}}$, one considers the provisional solution $\tilde{\mathbf{y}}$ as the input variable and $\tilde{\boldsymbol{\delta}}$ as the output variable, and defines an "implicit" function as

$$\tilde{\boldsymbol{\delta}} = \mathbf{H}(\tilde{\mathbf{y}}). \tag{110}$$

Notice that when $\tilde{\mathbf{y}}$ solves the GRK formulation, $\tilde{\boldsymbol{\delta}} = \mathbf{0}$. Therefore, instead of simply accepting the Neumann series solution as in the original SDC methods, in the KDC method, the Jacobian-Free Newton Krylov (JFNK) method [240] is applied to solve $\mathbf{H}(\tilde{\mathbf{y}}) = \mathbf{0}$, and each function evaluation of $\mathbf{H}(\tilde{\mathbf{y}})$ in the JFNK iteration is nothing more than one SDC iteration. Unsurprisingly, $\mathbf{H}(\tilde{\mathbf{y}}) = \mathbf{0}$ is better conditioned compared with the original collocation formulation since the low-order method provides an approximation of the collocation formulation.

The KDC scheme has also been generalized to solve PDAE systems as discussed in [90,211]. In this approach, discretization is first performed in the temporal direction using Gaussian nodes. For the resulting coupled elliptic equation system, similar to the KDC methods for ODEs and DAEs, a lower order time stepping method is used to precondition the system, and the JFNK method is then applied to find the zero of the better conditioned system. For the elliptic equation at each time step in the low-order method, an integral equation based fast elliptic equation solver can be applied if high accuracy solutions are required [e.g. 149,250]. The fast elliptic solver is often accelerated by fast convolution type matrix vector multiplication algorithms [166], or by fast direct inversion algorithms utilizing the "low separation rank" property in the linear system [165].

### 4.3.2. A geochemical system application

To illustrate the ideas and demonstrate the performance of the KDC scheme in a water resources application, we consider a simple instance of calcite dissolution which is of frequent concern in carbon capture and geological storage in geochemical systems [313,331]. There are eight species: the three primary species (indexed 1–3) are $Ca^{2+}$, $HCO_3^-$, and $OH^-$, and the secondary species (indexed 4–8) include $H^+$, $CO_2$, $H_2CO_3^*$, $CO_3^{2-}$, and $CaHCO_3^+$. We assume that the dissolution/precipitation is the only rate controlled reaction, cation exchange is neglected, and changes in the pore structure are considered insignificant. Furthermore, to simplify the discussion and focus on the KDC approximation we extract a local set of geochemical reactions in the form of a system of DAEs in the absence of flow and transport operators, such as would result from using an operator splitting approach. With these assumptions, the corresponding PDAE system becomes a (relatively) simple DAE system consisting of three differential equations and five algebraic constraints, which can be obtained by applying a preprocessing procedure to the original eight-species reaction system [101,153]. Specifically, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}(y_2+y_5+y_6+y_7+y_8)=r, \tag{111}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(y_2-y_3+y_4+2y_5+2y_6+y_8)=0, \text{ and} \tag{112}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(y_2+y_5+y_6+y_7-y_1)=0, \tag{113}$$

where $y_i$ is the concentration of species $i$ for $i=1,\ldots,8$ and $r$ describes the rate-limited precipitation and dissolution of calcite

$$r=(1-\Omega_s)/10. \tag{114}$$

In Eq. (114) $\Omega_s$ is a saturation index that is a measure of whether the aqueous phase is subsaturated or supersatured with respect to calcium and carbonate, and we have fixed the product of the rate constant and specific reactive surface area of calcite for simplicity.

The remaining five algebraic constraints are given by

$$y_6=(K_1+1)y_5, \tag{115}$$

$$y_6=\tilde{a}_2\tilde{a}_4/K_2, \tag{116}$$

$$\tilde{a}_2=\tilde{a}_7\tilde{a}_4/K_3, \tag{117}$$

$$\tilde{a}_8=\tilde{a}_1\tilde{a}_2/K_4, \tag{118}$$

$$\tilde{a}_3\tilde{a}_4=K_w, \tag{119}$$

where $K_i$ $(i=1,\ldots,4)$ and $K_w$ are relevant equilibrium constants, $\tilde{a}_i$ is the activity of species $i$ given by the relation $\tilde{a}_i=\hat{\gamma}_i y_i$, and the activity coefficient $\hat{\gamma}_i$ is a function of the ionic strength of the solution, computed using the extended DeBye-Hückel relation

$$-\log_{10}\hat{\gamma}_i=\frac{AZ_i^2\mu^{1/2}}{1+B\alpha_i\mu^{1/2}}. \tag{120}$$

In the formula,

$$\mu=\frac{1}{2}\sum_{i\in\mathcal{I}_s}y_iZ_i^2, \tag{121}$$

$\mathcal{I}_s$ is the index set of species, $Z_i$ is the charge of species $i$, and $A$ and $B$ are properties of the solvent that depend upon temperature. Lastly, the saturation index is given by

$$\Omega_s=\frac{\tilde{a}_1\tilde{a}_7}{K_{sp}}, \tag{122}$$

where $\Omega_s=1$ at equilibrium, $\Omega_s<1$ when the aqueous phase is subsaturated with respect to calcite, and $\Omega_s>1$ when the aqueous phase is super-saturated with respect to calcite. Values for the various constants are given in Table 1.

Using Picard type integral, we symbolically denote the DAE system as

$$\mathbf{F}\left(\mathbf{y}^0+\int_0^t\mathbf{Y}(\tau)d\tau,\mathbf{Y}(t),t\right)=\mathbf{0}, \tag{123}$$

where $\mathbf{Y}(t)=[Y_1(t),Y_2(t),\ldots,Y_8(t)]=\mathbf{y}'(t)=\frac{d}{dt}[y_1(t),y_2(t),\ldots,y_8(t)]$ is introduced as the new unknown function, $\mathbf{y}^0$ is the initial condition, and $y_i(t)$ can be recovered using Gaussian quadrature integration.

Introducing the spectral integration matrix $\mathbf{S}$ (which maps the function values at the Gaussian nodes to its integral at the same node points using orthogonal polynomial approximation and Gaussian integration, see [196]), the KDC scheme solves the discretized collocation (pseudospectral) formulation for Eq. (123) given by

$$\mathbf{F}(\mathbf{y}^0+\Delta t\mathbf{S}\otimes\mathbf{Y},\mathbf{Y},\mathbf{t})=\mathbf{0}, \tag{124}$$

where $\mathbf{t}=[t_1,t_2,\ldots,t_s]^T$ represents the Gaussian nodes, $\mathbf{y}^0$ is the vector of initial values, $\mathbf{Y}=[\mathbf{Y}_1,\mathbf{Y}_2,\ldots,\mathbf{Y}_8]^T$ with each vector $\mathbf{Y}_i$ the desired collocation formulation solution approximating $Y_i(t)$ at the discretized points $\mathbf{t}$, and $\otimes$ is the tensor product (i.e., $\Delta t\mathbf{S}$ is applied to each component of $\mathbf{Y}$). Clearly, the direct solution of the collocation formulation when $s$ is large is in general computationally inefficient as the matrix $\mathbf{S}$ is dense. Instead, in the KDC scheme, the deferred correction technique is applied to precondition Eq. (124). The resulting nonlinear system for the error's equation is solved using the JFNK method, as detailed next.

We first assume provisional solutions $\tilde{\mathbf{Y}}_i=[\tilde{Y}_i(t_1),\tilde{Y}_i(t_2),\ldots,\tilde{Y}_i(t_s)^T]$ at the nodes $\mathbf{t}$ are available (derived using a low order time stepping method or simply using the initial values), and define an equation for the error $\delta_i(t)=Y_i(t)-\tilde{Y}_i(t)$ by

$$\mathbf{F}\left(\mathbf{y}^0+\int_0^t\left(\tilde{\mathbf{Y}}(\tau)+\boldsymbol{\delta}(\tau)\right)d\tau,\tilde{\mathbf{Y}}(t)+\boldsymbol{\delta}(t),t\right)=\mathbf{0}, \tag{125}$$

where $\tilde{Y}_i(t)$ is the interpolating polynomial of $\tilde{\mathbf{Y}}_i$. Notice that the error equation is similar to the original system, therefore an existing *low-order method* can be adapted to obtain $\tilde{\delta}$ that approximates the error and improves the provisional solution $\tilde{\mathbf{Y}}_i$ for $i=1,2,\ldots,8$. In our implementation, for stiff DAE systems, a backward Euler method is applied when marching from $t_m$ to $t_{m+1}$ as in

$$\mathbf{F}\left(\mathbf{y}^0+[\Delta t\mathbf{S}\otimes\tilde{\mathbf{Y}}]^{(m+1)}+\sum_{l=1}^{m+1}\Delta t_l\boldsymbol{\delta}^{(l)},\tilde{\mathbf{Y}}^{(m+1)}+\boldsymbol{\delta}^{(m+1)},t_{m+1}\right)=\mathbf{0}, \tag{126}$$

**Table 1**
Parameter values for calcite dissolution example.

| Parameter | Value |
|---|---|
| $A^\dagger$ | $5.00\times10^{-1}$ |
| $B^\dagger$ | $3.26\times10^{-1}$ |
| $K_1$ | $1.58\times10^{-3}$ |
| $K_2$ | $3.80\times10^{-7}$ |
| $K_3$ | $3.72\times10^{-11}$ |
| $K_4$ | $5.50\times10^{-2}$ |
| $K_w$ | $4.57\times10^{-15}$ |
| $K_{sp}$ | $3.98\times10^{-9}$ |

Ion activity constants $\alpha_i$ are taken from [331] 15 °C [331].

where $\Delta t_{l+1} = t_{l+1} - t_l, t_0$ and $\delta^0 = [\delta_1(t_0), \delta_2(t_0), \dots, \delta_8(t_0)]$ are set to **0**. In matrix form, we have

$$\mathbf{F}(\mathbf{y}^0 + \Delta t\mathbf{S} \otimes \tilde{\mathbf{Y}} + \Delta t\tilde{\mathbf{S}} \otimes \tilde{\boldsymbol{\delta}}, \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}, \mathbf{t}) = \mathbf{0}, \tag{127}$$

where $\Delta t\tilde{\mathbf{S}}$ is the corresponding lower triangular representation of the rectangle rule approximation of the spectral integration operator $\mathbf{S}$. Specifically,

$$\Delta t\tilde{\mathbf{S}} = \begin{bmatrix} \Delta t_1 & 0 & \cdot s & 0 & 0 \\ \Delta t_1 & \Delta t_2 & \cdot s & 0 & 0 \\ \cdot & \cdot & \cdot s & 0 & 0 \\ \Delta t_1 & \Delta t_2 & \cdot s & \Delta t_{s-1} & 0 \\ \Delta t_1 & \Delta t_2 & \cdot s & \Delta t_{s-1} & \Delta t_s \end{bmatrix}. \tag{128}$$

Further notice that the approximate error $\tilde{\delta}$ can be considered as a function of the given provisional solution $\tilde{\mathbf{Y}}$ from the "implicit" function in Eq. (127), and when the "input" variable $\tilde{\mathbf{Y}}$ solves the original collocation formulation in Eq. (124), the "output" is $\tilde{\boldsymbol{\delta}} = \mathbf{0}$. Using the same notation as in Eq. (110) and applying the implicit function theorem, one can show that this new function $\tilde{\boldsymbol{\delta}} = \mathbf{H}(\tilde{\mathbf{Y}})$ is better conditioned (which is not surprising as the low order method solves a "nearby" problem), and existing JFNK methods [225,240] can be applied directly for its efficient solution. We omit the details of the JFNK approximation.

In Fig. 2, we compare the accuracy and efficiency of the KDC method with the DASPK package. The current KDC solver was implemented in Matlab, with options for either a fixed time step size or a simple adaptive marching strategy. To test the accuracy of the methods, we use a manufactured analytical solution $y_i(t) = \cos(i * t)$ by adding source terms to the original system. In (a), we consider a set of parameters $K_i$ $(i = 1, \dots, 4), K_w$, and $K_{sp}$ such that the DAE system is non-stiff. Specifically, we set $K_1 = 1.58,\ K_2 = 3.80,\ K_3 = 3.72,\ K_4 = 4.57$, and $K_{sp} = 3.98$. For the KDC method, we plot the number of nonlinear solves for different choices of step sizes (uniform step sizes are used) and number of nodes $s = 3, 4, \dots, 11$. For DASPK, we plot results for different error tolerances. In DASPK, we count each attempted marching step as one nonlinear solve (plus any nonlinear solver failures), and in KDC, one nonlinear solve is required when marching from $t_k$ to $t_{k+1}$ using the backward Euler method. In (b), we consider the original set of parameters $K_i$, $K_w$, and $K_{sp}$ for the physical system (given in Table 1), numerical calculation shows that the Jacobian matrix has eigenvalues with large imaginary part in this setting. For both cases, we march from $t = 0$ to $t = 0.4$. Numerical results show that for the non-stiff case and for low accuracy requirements (3 to 4 significant digits), DASPK outperforms KDC. However, as the solutions are smooth, the KDC accelerated collocation formulation allows larger time step sizes for the same accuracy requirement, and hence is more efficient for higher accuracy requirements. For the stiff case, due to stability region properties, DASPK becomes less efficient when compared with KDC even for low accuracy requirements (see (b)). Also, we noticed that for both stiff and non-stiff systems and for the same accuracy requirements, the KDC methods using more node points (higher order) are more efficient than lower order KDC methods with smaller time step sizes, which is not surprising as the KDC method indeed solves the pseudo-spectral formulation in one big time step.

Finally, we want to mention that as an evolving numerical scheme, the KDC time stepping scheme requires further detailed study in order to better understand its applicability and numerical properties. First, for non-smooth data, the convergence of the spectral or pseudo-spectral discretization will deteriorate. In many water resources models, such discontinuities or sharp fronts may require special treatment in order to make the SDC or KDC accelerated PDAE solver applicable. Second, it was also observed and analytically validated that an "artificial" boundary layer will appear in the elliptic equation system after temporal discretization of the PDAE, whose magnitude is bounded by the temporal discretization error. It is therefore unnecessary for the elliptic equation solver to resolve this layer to an accuracy higher than necessary. This phenomenon was referred to as the "controlled stiffness" in [359]. Compared with the stiffness for ODEs, the boundary layer presents a different type of challenge for existing elliptic equation solvers. Last but not least, in order to fully explore the efficiency of KDC methods, the current MATLAB based KDC solver needs to be optimized. In particular, we are studying strategies for better selections of adaptive step-size, order of the method, proper Newton–Krylov methods, simplified Newton approximations, semi-implicit low-order preconditioners, and parameter choices within the algorithm.

### 4.4. Other notable, evolving discretization methods

While it is clearly not possible to give them full attention, or even fully enumerate them, there are other spatial discretization methods that have seen considerable interest over the last ten years. For example, the multipoint flux approximation method (MPFA) is an extension of block-centered finite difference methods for Darcy flow problems to non-rectangular and, more generally, non-orthogonal grids [47,131,132,284]. The basic idea behind these methods is to preserve the properties of block-centered finite

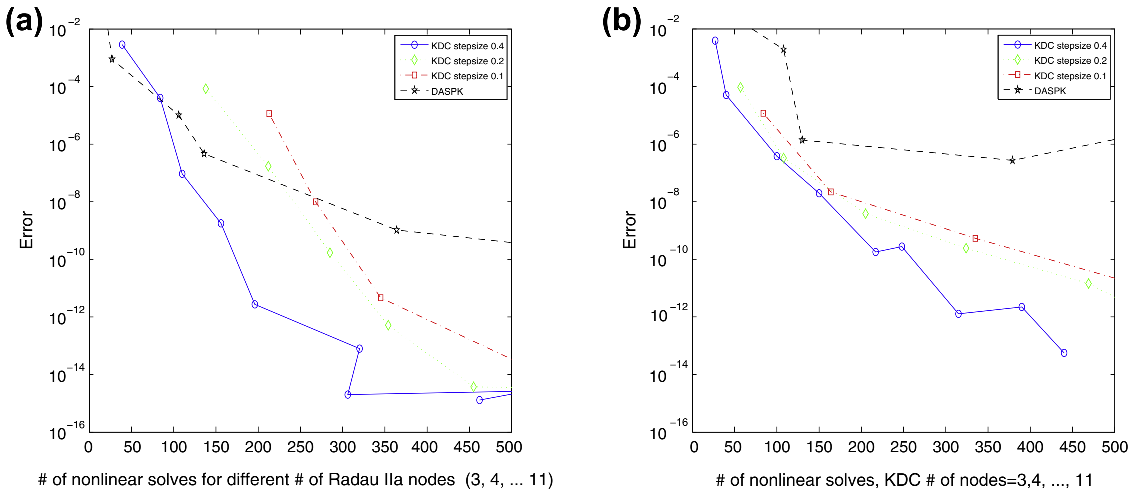

**Fig. 2.** Accuracy and efficiency comparison of the KDC method and DASPK for (a) non-stiff and (b) stiff settings.

differences: (1) the pressure unknown is defined at the center of the element, (2) the normal Darcy fluxes are continuous across element boundaries, (3) the method is locally mass conservative, and (4) the fluxes can be eliminated in terms of the pressure variable only. In addition to these properties, the pressure and fluxes should be at least globally first-order accurate, although in some cases second-order superconvergence can be proven at certain points (i.e., super-convergence points) in the mesh. MPFA methods have been under development for nearly two decades; we refer to [46,235] for discussion of the early developments of the MPFA methods and their relationship to other discretization schemes like control-volume finite element and mixed finite element methods. Since that time, interest and research into these methods has continued to develop. Closely related approaches like mimetic finite differences [86,93] and multipoint-flux mixed finite element methods [351] have shown promise as well.

## 5. Modern algebraic solution approaches

Many of the algorithmic descriptions in this section are taken from [224,225], both of which cover theory and implementation in more detail than we do here.

Discretization (see Section 4) produces a finite-dimensional linear or nonlinear system of equations. Our discussion of solvers is aimed at these kinds of applications. We will let $\mathbf{u} \in R^N$ denote the vector of $N$ unknowns. For example, after discretizing Eq. (1) in three space dimensions on an $n \times n \times n$ mesh, the vector $\mathbf{u}$ could be the nodal values of $H$ with $N = n^3$. If there are also $n_S$ reacting chemical species described nonlinearly in the flow, the system of algebraic equations is nonlinear and there are $1 + n_S$ degrees of freedom for each of the $N = n^3$ spatial mesh points.

Solver technology and computing power have evolved to the point where substantial problems in one or two space dimensions can be solved on laptop computers using environments like MATLAB and Python. Problems in three space dimensions, as well as many real-world applications in two space dimensions, often require implementations that exploit distributed-memory parallel computers and more specialized software environments. Such problems require parallel computing to meet both the memory requirements of the models and to return an accurate solution in a reasonable amount of wall clock time. Precise notions of *scalability* are useful for evaluating solver technology for parallel computing.

To fix ideas, we describe a simple abstract model of a parallel machine and application to the solution of problems in water resources. A parallel machine consists of $P$ processors, each with a fixed amount of memory and capable of a fixed rate of floating point operations per second. A typical algebraic system encountered in water resources modeling arises from a discretization of a PDE and results in $N$ computational degrees of freedom, with each degree of freedom requiring a fixed amount of memory. Assume we are able to solve the $N$ degree of freedom problem on a set of $P$ processors and find that the wall clock time required is $T$. Two common problems are that the computation ran too long ($N$ is acceptable but $T$ is too large) or the solution was not accurate ($N$ is too small but $T$ is acceptable). Resolving these problems requires that we have some estimate of how the computation *scales* as $P$ and possibly $N$ are increased. In the first scenario we evaluate "how $T$ varies as $P$ increases given fixed $N$", and the solver is said to be *strongly scalable* if wall clock time $T$ decreases linearly as the number of processors $P$ increases linearly. In the second scenario we evaluate "how $T$ changes as $N$ increases for a fixed ratio $N/P$", and the solver is said to be *weakly scalable* if the wall clock time $T$ is fixed as $N$ increases. This terminology is odd in the sense that weak scalability is the more critical notion for most challenging applications in water resources and requires careful implementation and numerical analysis to achieve. After all, computational water resources is primarily concerned with expanding the accuracy, scales, and range of processes—thus increasing the degrees of freedom of the models.

An issue related to weak scalability is how a method performs as the discretization is refined. For example, one might show that an iterative method can reduce the residual at a rate that does not depend on the spatial mesh width. We will refer to this independence of convergence on the discretization as mesh-independence. Mesh-independent algorithms are a necessary, but not sufficient, condition for scalable parallel performance, since many additional considerations other than operation counts alone determine parallel performance.

### 5.1. Linear equations

We will write linear systems as

$$\mathbf{Au} = \mathbf{f}. \tag{129}$$

We will assume throughout that Eq. (129) has a unique solution for all right-hand side vectors $\mathbf{f}$. We will think of $\mathbf{A}$ as a linear operator, rather than a matrix, because some methods (the matrix-free algorithms) only need the product of $\mathbf{A}$ with a vector, which is often much less expensive to compute than a matrix representation of $\mathbf{A}$.

We distinguish between direct methods, which in infinite precision will return the solution in finitely many operations, and iterative methods, which produce a sequence of approximations to the solution.

The performance and accuracy of linear solvers depends on the condition number of the operator

$$\kappa_c(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|. \tag{130}$$

If, for example, $\kappa_c(\mathbf{A})$ is very large ($>10^8$) then the residual norm $\|\mathbf{f} - \mathbf{Au}\|$ can be a poor indicator of the error, and any solver can produce poor results [122,224].

#### 5.1.1. Direct methods

Even in one space dimension, the discretizations of interest for water resources models can have too many unknowns to use simple Gaussian elimination [122] without exploiting the sparsity of the problem. Sparse direct solvers [114] apply Gaussian elimination using the pattern of non-zeros in the coefficient matrix. These methods must compute and store the matrix representation of the discrete operator, and then perform a sparse matrix factorization. This cost is often prohibitive in three space dimensions. However in one or two dimensions, these methods are often practical and efficient. Table 2 gives a list of software implementing direct solvers. Using sparse solvers efficiently typically only requires that a user provides the matrix in a sparse format—most solvers are designed to have an application program interface (API) quite similar to dense direct solvers. To extract optimal performance and robustness additional algorithm-dependent parameters may also require tuning (e.g. approximating required fill, drop tolerance, etc.). The packages in Table 2 are written in Fortran, C, and C++. Some of these sparse direct solvers are also conveniently available in MATLAB and Python (http://www.mathworks.com/products/matlab, scipy.sparse.linalg).

#### 5.1.2. Iterative method fundamentals

For problems in three spatial dimensions, iterative methods are a compelling choice. However, iterative methods require more programming, design, and analysis. The most fundamental new requirement relative to sparse direct solvers is that termination criteria need to be selected appropriately for the application in or-

**Table 2**
Parallel software for sparse direct solvers.

| Name | URL | References |
|---|---|---|
| Amesos | http://trilinos.sandia.gov/packages/amesos | [314,315] |
| SuperLU | http://crd-legacy.lbl.gov/~xiaoye/SuperLU | [123,124,255] |
| UMFPACK | http://www.cise.ufl.edu/research/sparse/umfpack | [112,113,115,116] |
| MATLAB | http://www.mathworks.com/products/matlab | [188] |
| MUMPS | http://graal.ens-lyon.fr/MUMPS | [51] |
| PARADISO | http://www.pardiso-project.org | [319,320] |
| SPOOLES | http://www.netlib.org/linalg/spooles/spooles.2.2.html | [57] |
| PSPASES | http://www.cs.umn.edu/~mjoshi/pspases | [171] |
| TAUCS | http://www.tau.ac.il/~stoledo/taucs/ | [307] |

der for iterative solvers to be efficient. Selecting termination criteria can be quite complex in practice as it involves balancing accuracy with performance and robustness of the overall computation, which may include feedback from time step selection and nonlinear iteration performance. There are many other considerations for the large family of existing iterative methods including generic considerations such as efficient parallel sparse matrix–vector products as well as algorithm-specific details such as restart strategies for methods that require the preservation of a sequence of solutions over multiple iterations, which we will refer to as iteration history.

In stationary solution methods, it is common to terminate iterative methods when the relative residual is sufficiently small. Given a tolerance $\eta$, the termination criterion is

$$\|\mathbf{f} - \mathbf{Au}\| \leqslant \eta\|\mathbf{f} - \mathbf{Au}^0\|, \tag{131}$$

where $\mathbf{u}^0$ is the initial guess of the solution, which is often taken as zero. In time-dependent problems, the solution at the previous time step, or an extrapolation of it, may provide a very good $\mathbf{u}^0$. Then a relative criterion can be too strict, and the absolute residual is often more relevant for terminating the iterations:

$$\|\mathbf{f} - \mathbf{Au}\| \leqslant \eta. \tag{132}$$

Here, the tolerance $\eta$ must match the characteristic size of elements in $\mathbf{u}$. To allow the flexibility of both relative and absolute criteria many implementations require a relative tolerance $\eta_r$ and an absolute tolerance $\eta_a$ and terminate the iterations when

$$\|\mathbf{f} - \mathbf{Au}\| \leqslant \eta_a + \eta_r\|\mathbf{f} - \mathbf{Au}^0\|. \tag{133}$$

In many iterative methods, it is natural to terminate based on a preconditioned residual, which can be viewed as an approximation of the error, that is

$$\|\mathbf{u} - \mathbf{u}^*\| \approx \|\mathbf{Pf} - \mathbf{PAu}\| \leqslant \eta_a + \eta_r\|\mathbf{Pf} - \mathbf{PAu}^0\|. \tag{134}$$

where $\mathbf{u}^*$ is the exact solution vector, and $\mathbf{P}$ is the preconditioner matrix. Clearly the preconditioner and the tolerances must be carefully chosen in order for this test to yield reliable and/ or efficient iterative methods.

### 5.1.3. Stationary iterative methods

Stationary iterative methods convert Eq. (129) to a linear fixed point problem

$$\mathbf{u} = \mathbf{Mu} + \mathbf{b}, \tag{135}$$

where $\mathbf{M}$ is called the iteration matrix. The iteration is

$$\mathbf{u}_{n+1} = \mathbf{Mu}_n + \mathbf{b}. \tag{136}$$

The classical stationary iterative methods use a matrix representation of $\mathbf{A}$ to construct $\mathbf{M}$. More modern (and more efficient) stationary iterative methods, such as multigrid methods [87], do not necessarily require a matrix representation of $\mathbf{A}$. Stationary iterative methods converge if the largest eigenvalue of $\mathbf{M}$ is smaller than 1 in magnitude, and their analysis is based on an estimate of that eigenvalue [87].

The prototypical stationary iterative method is the Jacobi iteration [122,224]. This method requires a matrix representation of $\mathbf{A}$, and solves the $i$th equation in Eq. (129) for $u_i$ to obtain the iteration

$$(u_n)_i = \frac{1}{a_{ii}}\left(f_i - \sum_{j\neq i} a_{ij}(u_{n-1})_j\right), \tag{137}$$

where $(u_n)_i$ is the $i$th coordinate of iteration $n$. For this example, if we split $\mathbf{A}$ into diagonal and strictly upper and lower triangular parts

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U} \tag{138}$$

then

$$\mathbf{b} = \mathbf{D}^{-1}\mathbf{f} \quad \text{and} \quad \mathbf{M} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}). \tag{139}$$

The classical stationary iterative methods perform poorly (in terms of the number of floating point operations needed to produce a solution of a given accuracy) for discretizations of differential equations, and the performance becomes worse as one refines the grid [122]. However, they can be very efficient in terms of storage and can, for some problems (like finite difference discretizations of differential equations), be implemented in a matrix-free way. These methods are also the basis for some useful preconditioners.

The performance of iterative methods is often a function of the condition number of the matrix. Before moving onto Krylov subspace methods and preconditioning, it is useful to provide a more precise statement of why performance of classical stationary iterative methods perform poorly as one refines the grid of a discrete approximation of a PDAE, which implies that they are not weakly scalable. We take as an example the linear, steady-state version of the advection–diffusion equation, Eq. (2), in a three-dimensional domain $\Omega$. We use the shorthand $L(u)$ for the linear differential operator applied to the solution $u$. After discretizing this equation with any standard method we obtain a linear algebra problem of the same form as Eq. (129) with the matrix $\mathbf{A}$ being the discrete approximation to the operator $L$. Discretizations of $L$ based on localized approximations (i.e., finite elements with locally supported basis functions and finite difference/volume methods) have the property that [144]:

$$\kappa_c(\mathbf{A}) \leqslant ch_e^2, \tag{140}$$

where $c$ is some constant independent of the discretization and $h_e$ is a parameter describing the grid, for example, the maximum cell length scale. This upper bound is asymptotically correct and implies that mesh independence (more generally weak scalability) cannot be attained for any iterative method with a convergence rate depending on the condition number. The essential problem is connected with the properties of the elliptic linear operator $L$: it has an unbounded set of eigenvalues, and the Green's functions have global support [150]. The discretization must either include globally supported approximations, resulting in dense matrices $\mathbf{A}$, which result in large storage and expensive matrix–vector products, or one must devise iterative algorithms that are independent of the condition number [144]. In the direction of iterative algorithms are the geometric and algebraic multigrid methods (as solvers, or as preconditioners for Krylov methods), which achieve mesh-independent convergence rates for a large class of discrete problems arising from the discretization of PDEs. Practical applications of multigrid methods have demonstrated both weak scalability and robust convergence for a wide range of applications, but this improvement comes at the cost of increased algorithmic complexity [172].

### 5.1.4. Krylov subspace methods

Krylov methods are not stationary iterative methods. The *k*th iteration will minimize some measure of error over the *k*th shifted Krylov subspace

$$\mathcal{K}_k = \text{span}(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0) \tag{141}$$

for $k \geqslant 1$. Here $\mathbf{r}_0 = \mathbf{f} - \mathbf{A}\mathbf{u}_0$ is the initial residual vector.

There are many Krylov methods. Complete theory exists for the conjugate gradient (CG) [185] method, which is designed for symmetric positive definite **A**, as you will get for example from the discrete solution of Eq. (1), and GMRES [310], which is for general operators. These methods differ in the measure of error that exists after a given iterate [224].

The *k*th CG iteration minimizes $\|\mathbf{u} - \mathbf{u}^*\|_A$ over $\mathbf{u}_0 - \mathcal{K}_k$, where $\mathbf{u}^* = \mathbf{A}^{-1}\mathbf{f}$ is the solution and the *A*-norm is defined by

$$\|\mathbf{u} - \mathbf{u}^*\|_A = \sqrt{(\mathbf{u} - \mathbf{u}^*)^T \mathbf{A} (\mathbf{u} - \mathbf{u}^*)}. \tag{142}$$

The *k*th GMRES iteration minimizes $\|\mathbf{f} - \mathbf{A}\mathbf{u}\|_2$ over $\mathbf{u}_0 - \mathcal{K}_k$. We will not discuss implementation here and, particularly for any method more complex than CG, encourage the reader to use one of the many Krylov solvers written by experts [66,183,224] or those included, for example, in MATLAB. CG codes terminate the iteration when the $l^2$ residual norm is sufficiently small, and hence the termination criterion is not coupled to the minimization problem the Krylov iteration solves. This is not a problem for the differential equation problems of interest here.

A very useful heuristic, which can be justified for CG and GMRES, is that Krylov methods perform very well when **A** is close to the identity matrix or if **A** has a few small clusters of eigenvalues (and is diagonalizable). One can also say that if $\kappa_c(\mathbf{A})$ is even moderately large ($>10^3$) the Krylov methods will converge very slowly. Discretizations of differential operators lead to poorly conditioned linear systems, and one must apply a preconditioner (see Section 5.1.5) to make Krylov methods perform well.

CG is very efficient in terms of storage, needing only five vectors for the entire iterative method, plus any storage used for **A**. GMRES, on the other hand, must store a basis of *k* orthonormal vectors for the Krylov subspace, and for very large problems, it can be impossible to meet this demand. This requirement basically means that to preserve the convergence proven for GMRES, one must store an additional basis vector for each iterate and use the entire stored basis in operations needed to produce the $k+1$ approximation—leading to a linear growth in both storage and operations needed for each successive GMRES iterate. Low-storage alternatives to GMRES attempt to address this, but all have limitations.

One might think that replacing $\mathbf{A}\mathbf{u} = \mathbf{f}$ with $\mathbf{A}^T\mathbf{A} = \mathbf{A}^T\mathbf{f}$ and using CG would work. However, this approach, called CGNR [224] squares the condition number of the discrete problem, and hence dramatically reduces the convergence rate, and it also requires two matrix–vector products per iteration, with one transpose-vector product. Methods like CGNR are too inefficient for the computational water resources models considered here.

Other non-symmetric Krylov solvers, such as Bi-CGSTAB [342] and TFQMR [156], have fewer problems. Neither of these methods requires a transpose-vector product, but both do need two matrix–vector products and neither has a complete convergence theory. Bi-CGSTAB and TFQMR can, and do, completely fail in certain situations. So the user of these methods must be prepared to do things like reinitialize a failed iteration. Finally, one may consider limiting GMRES to an *m*-dimensional Krylov subspace, and then restarting the iteration when the storage is exhausted. This method is called GMRES(m), and most GMRES codes allow one to limit the storage in this way. GMRES(m) does not share the rigorous convergence theory of full GMRES, and can fail to converge, but also performs well in many cases typical of the applications considered here.

### 5.1.5. Preconditioning

A preconditioner for Eq. (129) is an operator **P** that is inexpensive to apply to a vector and for which **PA** or **AP** converges more economically than Krylov solvers based on **A**. If **A** is symmetric and positive definite and one wants to use CG, then **P** must also be symmetric and positive definite. The preconditioned conjugate gradient method (PCG) use matrix–vector products and preconditioner–vector products to replace Eq. (129) with

$$\mathbf{P}^{1/2}\mathbf{A}\mathbf{P}^{1/2}\mathbf{y} = \mathbf{P}^{1/2}\mathbf{f}, \tag{143}$$

and solves Eq. (143) with CG, and then recovers $\mathbf{u} = \mathbf{P}^{1/2}\mathbf{y}$. Here $\mathbf{P}^{1/2}$ is the unique symmetric positive definite square root of **P**. PCG does not compute a matrix square root, and the solve of Eq. (143) is done indirectly [122,224].

For non-symmetric methods, one can apply **P** to either the right or the left of **A** without having to worry about maintaining symmetry. Left preconditioning replaces Eq. (129) with

$$\mathbf{P}\mathbf{A}\mathbf{u} = \mathbf{P}\mathbf{f}, \tag{144}$$

and applies the Krylov method to Eq. (144). Right preconditioning replaces Eq. (129) with

$$\mathbf{A}\mathbf{P}\mathbf{y} = \mathbf{f}, \tag{145}$$

applies the Krylov method to Eq. (145), and recovers $\mathbf{u} = \mathbf{P}\mathbf{y}$ after the solve. Left preconditioning has the advantage that the preconditioned residual will be used to control the iteration. If **PA** is better conditioned than **A**, then the preconditioned residual should be a better predictor of error. Right preconditioning has $\mathbf{f} - \mathbf{A}\mathbf{P}\mathbf{y} = \mathbf{f} - \mathbf{A}\mathbf{u}$ for the residual, which is the same as the unpreconditioned residual. This is an advantage in cases where reducing the residual of the discrete problem is the objective or when one wants to compare preconditioners.

The simplest preconditioner is Jacobi or diagonal preconditioning. Here $\mathbf{P} = \mathbf{D}^{-1}$, the inverse of the diagonal part of **A**. It is often possible to extract the diagonal entries of **A** from the discretization directly, without building a matrix representation of **A**. Other classical stationary iterative methods usually provide more efficient preconditioners, especially if formulated in a block version, where the unknowns are grouped in blocks, and where all unknowns inside a block are treated implicitly. Such iteration methods are closely related to domain decomposition. Typically, the domain for the PDAE is partitioned into subdomains and the problem is solved on each subdomain. One can show that such a strategy is a good starting point for an effective preconditioner for Krylov methods [327]. Domain decomposition methods can be formulated either via the physical domain for the PDAE or by appropriate partitioning of a matrix representation of **A**. Many domain decomposition methods are nearly mesh-independent and scale well. Examples of successful application of domain decomposition techniques can be found across the water resources field [207,208,219,339]. The PETSc package [69] was originally developed for research on domain decomposition methods and offers basic functionality for utilizing such methods.

Choosing $\mathbf{P}^{-1} = \mathbf{L}\mathbf{U}$, where **LU** is the lower–upper decomposition factorization of the matrix representation of **A**, yields a preconditioner that is optimal in the sense that $\mathbf{P}\mathbf{A} = \mathbf{I}$, and Krylov solvers would converge in one iteration. However, **L** and **U** are much less sparse than the matrix representation of **A**, and therefore expensive and memory intensive to compute; computing **L** and **U** is equivalent to solving $\mathbf{A}\mathbf{u} = \mathbf{f}$ by a sparse direct method. One idea is to drop all fill-in generated by the factorization procedure such that **L** and **U** are as sparse as the original coefficient matrix. These approximate **L** and **U** factors provide a preconditioner, called incomplete LU factorization (ILU). The method can also allow a certain degree of fill-into improve the preconditioner. Unfortunately, the ILU preconditioner is not mesh-independent, and it cannot be parallelized well. However, for problem sizes relevant on serial computers the method is trivial to apply, mostly with robust behavior and significant speed up of Krylov solvers.

If the high-order differential term in the PDAE model is simple enough, it may be much easier to write a solver (multigrid, for example) for the high-order term than to write one for the entire equation. Preconditioning with a solver for the high-order term is, if one gets the boundary conditions right, in many cases provably mesh-independent [267]. Even a single multigrid cycle for the high-order term will suffice in many cases. Multigrid preconditioning based on the higher-order term is an example of so-called *physics-based* preconditioning where expert knowledge of the structure of the problem is leveraged to design effective preconditioners. Knowledge of the modeling objectives and the relative importance of both temporal and spatial differential operators for the problem can sometimes yield a simple and effective preconditioning strategy, but the range of approaches is beyond the scope of this article. We note that many of the solver frameworks available, and cited below, include explicit support for block and incomplete factorizations schemes, approximation of the Schur complement, and multilevel and domain decomposition strategies for tailoring physics-based solvers. One application of the latter type of preconditioning, applied to poroelastic models, appears in [174] using a high-level supporting software framework [173].

### *5.2. Nonlinear equations*

We will write systems of nonlinear algebraic equations as

$$\mathbf{F}(\mathbf{u}) = 0. \tag{146}$$

The default recommendations for a solution strategy are based on Newton's method. In its simplest form Newton's method is a four-step process from a current iteration $\mathbf{u}_n$ to the next iterate $\mathbf{u}_{n+1}$. One typically terminates the nonlinear iteration when

$$\|\mathbf{F}(\mathbf{u}_n)\| \leqslant \tau_a + \tau_r\|\mathbf{F}(\mathbf{u}_0)\|, \tag{147}$$

where $\mathbf{u}_0$ is the initial iterate and the relative and absolute tolerances $\tau_r$ and $\tau_a$ play the role that $\eta_r$ and $\eta_a$ do in Eq. (133).

The steps are

- **Evaluate:** Compute $\mathbf{F}(\mathbf{u}_n)$ and test for termination.
- **Evaluate:** Compute $\mathbf{F}'(\mathbf{u}_n)$.
- **Step:** Solve (or approximately solve) the linearized problem $\mathbf{F}'(\mathbf{u}_n)\mathbf{s} = -\mathbf{F}(\mathbf{u}_n)$.
- **Update:** Set $\mathbf{u}_{n+1} = \mathbf{u}_n + \mathbf{s}$.

Here $\mathbf{F}'(\mathbf{u}_n)$ is the Jacobian matrix of $\mathbf{F}$ at the point $\mathbf{u}_n$. Variations of Newton's method differ in the way one solves the linearized problem and related modifications (relaxation) of the Newton step, $\mathbf{s}$. The basic Newton iteration is usually written as

$$\mathbf{u}_{n+1} = \mathbf{u}_n - \mathbf{F}'(\mathbf{u}_n)^{-1}\mathbf{F}(\mathbf{u}_n). \tag{148}$$

The theory for Newton's method says that if the initial iteration $\mathbf{u}_0$ is near a solution $\mathbf{u}^*$ for which $\mathbf{F}'(\mathbf{u}^*)$ is nonsingular, and $\mathbf{F}'$ is Lipschitz continuous, then the iteration converges to $\mathbf{u}^*$ and the convergence is fast. By fast we mean q-quadratic [224],

$$\mathbf{e}_{n+1} = O(\|\mathbf{e}_n\|^2), \tag{149}$$

where $\mathbf{e}_n = \mathbf{u}_n - \mathbf{u}^*$ is the error in the solution for iteration level $n$. Eq. (149) means that the number of significant figures of accuracy will roughly double with each iteration, at least up to the limits of floating point accuracy.

If one solves the linearized problem with a direct method, a costly part of the algorithm is evaluating and factoring the Jacobian, while for iterative methods construction of a preconditioner can likewise be costly. Evaluating the Jacobian itself and solving the linearized problem (back substitution or application of an iterative method) represent other significant expenses that must be weighed in constructing an optimally efficient nonlinear solver.

If $\mathbf{F}$ is expensive to evaluate and one approximates the Jacobian with a forward difference, at a cost of $N$ additional calls to the function computing $\mathbf{F}$, the Jacobian computation can be far more costly than the matrix factorization. For discretizations of PDEs computing, $\mathbf{F}'$ is often comparable to the cost of computing $\mathbf{F}$, so the choice of the optimal balance of Jacobian updates and factorizations or formations of preconditioners must be determined on a case-by-case basis.

Most nonlinear solver codes give users the option of computing and factoring the Jacobian less frequently. The price for this is that the solution will usually take more iterations, but generally those iterations are much less expensive. The chord method only computes and factors $\mathbf{F}'$ at the initial iterate, so the iteration is

$$\mathbf{u}_{n+1} = \mathbf{u}_n - \mathbf{F}'(\mathbf{u}_0)^{-1}\mathbf{F}(\mathbf{u}_n). \tag{150}$$

The convergence of the nonlinear iteration is not as fast as for Newton, with the rate given as

$$\mathbf{e}_{n+1} = O(\|\mathbf{e}_0\|\|\mathbf{e}_n\|), \tag{151}$$

but the reduction in cost per iteration may increase the overall efficiency compared to a full Newton solve. Many implicit temporal integration codes [88,291] carry this idea further, and keep Jacobian information for several time steps.

Newton-iterative methods solve the linearized problem with an iterative method, and the preconditioning issues are the same as for linear problems, with the added question of how often to compute preconditioning information. Typically one terminates the linear solver when the relative residual is small, that is

$$\|\mathbf{F}'(\mathbf{u}_n)\mathbf{s} + \mathbf{F}(\mathbf{u}_n)\| \leqslant \eta_n\|\mathbf{F}(\mathbf{u}_n)\|. \tag{152}$$

Note that there is no absolute error test in Eq. (152). The reason is that the absolute termination test is applied in the nonlinear iteration before invoking any linear solve. The linear solver may use the tolerances for the nonlinear solver in computing $\eta_n$ (see [224] for an example). In some ODE and DAE codes, for example DASPK [85,266], the linear solver inherits both a relative and absolute error tolerance from the nonlinear solver, which, in turn, is inherited from the local truncation bounds imposed by the integrator.

In the context of nonlinear solvers, the termination tolerance $\eta_n$ is called the forcing term. The convergence of the nonlinear (or outer) iteration depends on the tolerance $\eta_n$ one imposes on the linear (inner) iteration:

$$\mathbf{e}_{n+1} = O(\eta_n\|\mathbf{e}_n\| + \|\mathbf{e}_n\|^2). \tag{153}$$

So, making $\eta_n \to 0$, for example, will make the nonlinear iteration converge rapidly, but at the cost of very expensive linear iterations. Most nonlinear solver software packages have a sophisticated forcing term selection algorithm [e.g. 138], and users can typically accept the choice in the code.

Most nonlinear solver codes use Krylov methods to solve the linearized problem. The Jacobian-vector product $\mathbf{F}'\mathbf{y}$ (for a vector $\mathbf{y}$) can then be approximated with a forward difference [66,183,224]: $\mathbf{F}'(\mathbf{u})\mathbf{y} = (\mathbf{F}(\mathbf{u} + \tilde{\delta}\mathbf{y}) - \mathbf{F}(\mathbf{u}))/\tilde{\delta}$ for some small $\tilde{\delta}$. This is a feasible approach if evaluating $\mathbf{F}$ is much faster than evaluating $\mathbf{F}'$, or if it is complicated and tedious to derive $\mathbf{F}'$ from $\mathbf{F}$. If the forward difference increment is chosen carefully, the performance of the iteration will not be affected by the approximation of the matrix–vector product.

The convergence theory we have discussed so far is local. This means that the theory requires that the initial iterate be near the solution. If the initial iterate is not near the solution, a line search may be used. For Newton's method one replaces Eq. (148) with [54]

$$\mathbf{u}_{n+1} = \mathbf{u}_n - 2^{-m}\mathbf{F}'(\mathbf{u}_n)^{-1}\mathbf{F}(\mathbf{u}_n), \tag{154}$$

where $m$ is the smallest non-negative integer such that the sufficient decrease condition holds

$$\|\mathbf{F}(\mathbf{u}_{n+1})\| \leqslant (1 - \alpha_l 2^{-m})\|\mathbf{F}(\mathbf{u}_n)\|. \tag{155}$$

Here $\alpha_l$ is an algorithmic parameter; $\alpha_l = 10^{-4}$ is standard. The theory for this algorithm is that if the iteration does not become unbounded and the smallest eigenvalue (in magnitude) of $\mathbf{F}'$ remains bounded away from zero, then $\{\mathbf{u}_n\}$ will converge to a solution at which the local convergence theory is valid. Line searches are a common way for nonlinear solver codes to manage convergence from arbitrary starting points, and more sophisticated line searches, such as the popular cubic line search in [294], are available.

Many nonlinear problems in water resources arise from quasi-linear differential operators, for example we may write the discrete PDE boundary value problem as

$$L(\mathbf{u}) = \mathbf{f}. \tag{156}$$

For such problems, a natural approach is a fixed point iteration of the form

$$\mathbf{u}_{n+1} = L^{-1}(\mathbf{u}_n)\mathbf{f}. \tag{157}$$

This approach can be viewed as "lagging" the nonlinearity in the operator $L$, and the characteristics of the resulting iterative methods, while typically unable to achieve the local quadratic convergence of the classical Newton method, can be attractive. For example, the method known as Picard iteration for the nonlinear Richards' equation for unsaturated flow has been reported to have more robust convergence for inaccurate initial guesses than Newton's method [82,289]. Lagging nonlinearities are also common in dealing with the Navier–Stokes equations due to the relatively simple quadratic nonlinearity arising from the inertial terms in the momentum equation. Nonlinear multigrid methods such as the full approximation schemes (FAS) can also be viewed as fixed-point iterations that do not require formation and/or inversion of the global Jacobian $\mathbf{F}'$, and which may achieve robust, mesh-independent convergence [217], though this class of methods has not been widely studied in the context of large-scale three-dimensional computational models. Recent work on Anderson acceleration for fixed-point iterations provides one route to improving on the slow rate of convergence of fixed-point methods [344]. Finally we note that another useful approach to nonlinear problems is applying Newton's method directly to the PDE before discretization. This approach can provide some additional flexibility and insight into the discrete approximations, $\mathbf{F}$, and $\mathbf{F}'$; for example multiscale methods and nonlinear iterations for Richards' equation have been derived using this formalism [222].

The final issue to address is non-uniqueness. The theory for Newton's method with a line search says that the iteration will either converge to a solution or fail in one of two ways, both easy to detect. The theory does not say that the solution Newton's method finds is physically significant. This issue can be partially addressed by continuation methods [159,223,226,317,346], which vary natural or artificial parameters to move from a known physically useful solutions to solutions of the equations with different values of the parameters.

### *5.3. Software for algebraic solution approaches*

Design and implementation of software for large algebraic systems is a significant undertaking that typically requires a dedicated team of developers and a large user base. Table 3 provides a listing of software packages for iterative methods and preconditioners. Many of these packages also include interfaces to direct solver packages.

**Table 3**
Parallel software for solving algebraic equations.

| Name | URL | References |
|---|---|---|
| PETSc | http://www.mcs.anl.gov/petsc | [67–69] |
| Trilinos | http://trilinos.sandia.gov | [183] |
| HYPRE | http://computation.llnl.gov/casc/software.html | [102] |
| pARMS | http://www-users.cs.umn.edu/saad/software/pARMS/index.html | |
| MTL4 | http://www.simunova.com/de/node/24 | [324] |
| PyAMG | http://code.google.com/p/pyamg/ | [81] |

Most of the software in Table 3 is written in Fortran, C, or C++ and is not easily accessible in user-friendly programming environments such as MATLAB. However, PETSc can be quite easily operated from Python with aid of the petsc4py package (http://code.google.com/p/petsc4py), and a similar interface, PyTrilinos (http://trilinos.sandia.gov/packages/pytrilinos/) also exists for Trilinos. PyAMG provides easy-to-use Python constructions for calling up Krylov solvers with algebraic multigrid preconditioning.

Direct methods and Krylov solvers are offered as "black boxes" by most software frameworks and are hence easy to apply. However, a Krylov solver needs a preconditioner, which must be provided by the user, either as a matrix or as an action $\mathbf{Pv}$ on some vector $\mathbf{v}$. Preconditioners based on classical stationary iterations are easy to construct directly from $\mathbf{A}$, algebraic multigrid preconditioners are to some extent available as "black boxes" (Hypre through PETSc, ML through Trilinos, and PyAMG), and ILU preconditioners are also offered by many linear algebra libraries. Other types of preconditioners may require substantial implementation work from a user.

## 6. General computational environments

The nature of scientific software implementations has changed considerably over the last few decades. During this period, the aim has been to create software that is *simple* (easy to use), *general* (can solve a wide set of physical problems if the mathematical formulations are similar), *efficient* (utilizes the hardware to run optimally fast), and *reliable* (a high-quality answer is guaranteed). These goals are unfortunately contradictory. The limited computing resources in the early years after the invention of the computer demanded a strict focus on efficiency and to a lesser extent reliability. Generality and simplicity were unaffordable properties, and much of the programming tradition that lasted for decades among scientists reflected this view. Reliable, well-tested codes required many years of development, resulting in software life-times on the scale of decades. These codes were also normally hard to modify and extend to new problems, thus slowing the speed of research.

A clear shift appeared in the 1990s with the increased popularity of C++ for scientific computing. The implementation techniques offered by C++, object-oriented programming and generic (template) programming, naturally encouraged generality in that common mathematics and numerics could more easily be "factored out" of the applications and offered in general, reusable libraries. For example, while older finite element packages were formulated in terms of a heat conduction element and a porous medium transport element, software favoring generality splits the physics and mathematics, where feasible, and casts a finite element as a geometric entity attached with a set of basis functions and degrees of freedom. This may allow one element in the library to be used for both heat conduction and porous medium transport.

Many C++ libraries of this type have appeared during the last two decades, mostly accompanied by a large set of diverse physical applications and by some sort of computational environment for easily implementing and exploring models. A non-exhaustive list of notable environments for finite element programming includes Diffpack [6,248], GetDP [10], Getfem++[11], FreeFEM++ [11], libmesh [17], deal.II [5], Sundance [36], FEniCS [261,262,279], Proteus [31,221], DUNE [7,155], and COMSOL Multiphysics [2]. Applications built on these packages are coded in C++, or (for some of the packages) alternatively in a high-level scripting language. Similar environments supporting finite volume methods are OpenFOAM [28], FiPy [9], Clawpack and its derivative PyClaw [83,227]. Two notable software projects aimed specifically at comprehensive water resources simulation while maintaining some aspects of the modularity and abstraction in the environments above include [126,241].

Many investigators in environmental sciences who intend to develop new models may benefit greatly from basing their coding on one of the mentioned comprehensive packages. Choosing the appropriate package is, unfortunately, a non-trivial task as there are major differences with respect to learning curves, numerical methods supported, existing application gallery, dependencies on third-party libraries, documentation, support for rapid prototyping, and performance on parallel computers.

Another major trend that gained widespread popularity in the 1990s was the use of *high-level languages* for scientific computing. The term high-level points to clear, compact syntax, close to the mathematical notation of the problem. One statement in a high-level language will typically reflect many statements in Fortran, C, and even in C++. Moreover, high-level languages usually support *dynamic typing*, which means that a variable is not declared with a special type and can therefore hold any type of object. This gives substantial programming convenience and flexibility. Popular high-level languages for scientific computations include MATLAB [20], Octave [25], R [33], IDL [16], Maple [18], Mathematica [19], Scilab [35], Sage [34], and Python [32]. MATLAB, Maple, Mathematica, Sage, and Python enable both symbolic and numerical computing.

High-level languages are usually very easy to learn, and technically much simpler to work with than C++. They are particularly well suited for rapid prototyping of new ideas as there is a comprehensive collection of standard numerical libraries that can be called to perform standard tasks such as linear system solution, eigenvalue computations, singular value decomposition, curve fitting, integration, optimization, and solving ordinary differential equations. Although code in high-level languages usually runs much slower than corresponding code in compiled languages (Fortran, C, and C++), the loss of efficiency might be minimal if the majority of the numerical computations are delegated to standard libraries, since these libraries feature highly optimized code in compiled languages. When some numerical operations must be hand-coded in the high-level language itself, one can resort to vectorization, which means replacing explicit loops over large arrays by an appropriate set of array operations. Each array operation will invoke highly optimized compiled code. Vectorization usually requires a rewrite of the loop-based computational algorithm.

MATLAB has gained a key position in the scientific community and simplified access to numerical computing for the masses of researchers and engineers. Python is an increasingly popular open source computing platform, having much in common with MATLAB, but is also a full-fledged programming language supporting all major programming styles, including "flat scripts" to procedural, object-oriented, generic, and functional programming. Contrary to MATLAB, plain Python is not suitable for scientific computing without a range of additional packages. By "Python" we shall mean the standard Python distribution *and* a collection of numerical packages—at least numpy [24], scipy [212], and matplotlib [200].

The support for developing PDE-based models in MATLAB and Python is limited. Finite difference schemes on structured grids are very straightforward to code, but the nested loops over large arrays can potentially lead to very slow code. For many investigations the speed may be acceptable, but usually special techniques must be invoked to increase performance. MATLAB has a just-in-time (JIT) compiler that typically can reduce the loss of performance down to a factor of 2–3 compared with hand-written Fortran, C, or C++. Python has an extension called Cython, which allows declaring variables with type and compiling the Python code to C. Finite difference schemes can then reach almost the speed of a hand-written C code. Vectorization in terms of displaced slices of arrays can also be utilized, resulting in significant speed-up, yet not comparable to what is gained by JIT compilation or Cython. Python supports migration of loops to native C, C++ or Fortran by various tools [249].

When it comes to finite element methods, there are numerous simpler codes in MATLAB and Python, typically developed for teaching purposes. Speed is also an issue for these applications, and it has proven to be difficult to match the performance and richness in methodology offered by the comprehensive C++ packages listed above. An ideal combination is to combine a high-level language with such a package. This is realized in the FreeFEM++and FEniCS projects. The former applies its own specialized high-level language, while FEniCS applies Python. With FEniCS, a finite element problem can be expressed in compact Python code, with a syntax that resembles the mathematical formulation. This code triggers a domain-specific compiler to generate corresponding C++ code, tailored to the problem, and linked to various compiled libraries (including PETSc and Trilinos) [231,232]. The resulting fast code is imported back in Python for execution. This use of code generation is a promising way to combine the ease and convenience of high-level languages and the high performance of compiled languages. The domain-specific compilation step may also feature optimizations that are hard or impossible to perform by hand [230,233].

FiPy and PyClaw are packages for finite volume discretizations, using Python as the interface language for setting up the problem and controlling the work flow. The combination of Fortran, PETSc, and Python in PyClaw has made it possible to conveniently define PDE problems in Python and still obtain very high performance and scalability with over 60,000 processors [228].

Reliability of scientific codes has traditionally been ensured by careful comparison of computed results with deep mathematical and physical insight into the problem at hand. Recently, the field of *a posteriori* error estimation has helped to implement error control thus improving the reliability in scientific codes. Moreover, error estimation enables optimal use of computation resources through adaptive meshing according to the estimated distribution of errors. The FEniCS project has taken this approach one step further with *automatic* generation of code for error estimation and error control, for any given (stationary) PDE model. This feature offers reliability and optimal use of resources to non-experts in the mathematics of error estimation and adaptivity [306].

To solve a PDE using one of the finite element packages discussed above a computational mesh is required. The level of sophistication of internally supported meshing varies, although Delaunay-based algorithms are often used for generating unstructured simplicial meshes. Widely used, stand-alone Delaunay triangulators include Triangle [323] and TetGen [37]. These tools can provide quality simplicial meshes in an automated way with user-specified quality measures. Even though Delaunay-based methods are applicable to complex domains there are classes of problems where semi-structured hexahedral or quadrilateral elements are preferable (e.g. for better grading in boundary layers). In this case, packages like Netgen [23], Cubit [3], and Gmsh [12] provide more comprehensive coverage of algorithms (e.g. advancing front techniques) and meshes with mixed element types. A long, but likely not exhaustive list of available software can be found at [22].

Many codes, like Triangle and Tetgen, are based on a command-line interface. While succinct, this interface is less than optimal in some cases. Some packages like Cubit and Gmsh provide a level of scripting support for building domains using geometric and topological primitives (e.g. using a boundary representation formalism) as well as an interface to various solid modeling software and triangulation techniques. Beyond this, they include a graphical interface with mesh and geometry creation and editing capabilities, which becomes more and more valuable as the complexity of the mesh generation problem increases and iteration with user input is needed to achieve necessary quality.

Mesh generation is a very broad field, and one that receives less attention in the water resources community than in other disciplines like mechanical or aerospace engineering. On the other hand, there are particular challenges in our field, like meshing fracture networks or resolving complex bathymetry and coastlines, that are non-trivial. Specific-purpose algorithms for such problems exist [269,281,321], but the availability of general, robust software is limited. We believe more attention is needed, and more progress could be made with the involvement of meshing expertise from the wider field. One such effort is the Department of Energy's Interoperable Technologies for Advanced Petascale Simulations (ITAPS) center [204], which aims to produce integrated software for mesh, geometry, and field manipulation for use directly in scientific applications.

The development of visualization in scientific computing has followed a similar arc as software for modeling PDEs. Consider a point of departure archiving simulation data for off-line visualization using tools like Gnuplot [13] or MATLAB. This approach is straightforward and for simple file formats can be handled easily using formatted I/O statements that are available in Fortran 77 or C. For more complex data formats like the XML-based XDMF [41] a higher level interface is preferable if not necessary. In any event, saving data to disk for off-line visualization works well when the solution data is not extremely large or when simulation turnaround times in a development phase are not too long. High-performance portable binary data formats such as HDF5 [14] or more domain specific formats that use it for array data (e.g. XDMF or NetCDF [283]) are recommended for extremely large data sets. The paradigm continues to be quite useful today, whether using Gnuplot or more sophisticated visualization environments and applications like Advanced Visualization Systems (AVS) [1], IBM's Data Explorer (now OpenDX) [27], EnSight [8], or ParaView [29] that have emerged since the 1990s.

For increasingly complicated problems and numerical algorithms, run-time visualization (either on screen or with off-screen rendering) has been recognized as a useful tool for development of more sophisticated numerical methods. To be efficient, some integration of graphical capabilities in a simulation code is required, which naturally increases code complexity. This integration is not particularly natural in a procedural Fortran 77 program but is more manageable using C or C++ given the availability of both low-level and high-level graphical libraries in these languages. An example at the time of the 25th anniversary AWR review can be seen in the patch-based, parallel AMR code from [339], which incorporated X11-based visualization using object-oriented data structures for multilevel grids and solution data.

While much more tractable using object-oriented techniques in C++, this approach requires fairly detailed understanding of graphical programming and tight integration into the simulation code. The evolution of higher level interpreted languages like Python has improved scientific visualization in both post-processing and run-time modes, since they can provide interfaces that are more intuitive and simpler to use for scientists and engineers. Two such examples are MayaVi [21] and matplotlib which, as a point of reference, is used for visualization in the recently developed GeoClaw AMR code from [83]. Moreover, a high-level language makes exploration of data in an interactive shell environment like IPython [290] much easier.

Visualization for distributed applications is obviously more challenging, but the emergence of VTK [40] and VTK-based software like ParaView and ViSit [39] has made this more approachable both for integrating visualization into a simulation code or in post-processing large, distributed data sets. For extremely large problems with terabytes of data, straightforward visualization of solution data becomes impractical to impossible. Rather, co-processing, or *in situ* visualization, [30,151] is emerging as a preferable technique. Here, data is not just extracted during a simulation for (off-screen) rendering. Using high-level tools like ParaView or VisIt, rendered views can be composited pipelines that provide targeted data representations (e.g. streamlines, multiple cut planes, iso-volumes, etc). These views can also be controlled by the user during the simulation.

## 7. Hardware trends and implications

Even today's small laptops allow a wide range of interesting PDE-based models in science and engineering to be computationally analyzed within feasible execution times. In this sense, serial computing on cheap, personal devices has become the most widespread and important way to explore such models. On the other hand, there has always been a drive among scientists and engineers to target the most complex models and the highest possible resolution that the most powerful accessible computers can manage. Supercomputers and institutional clusters have, during the last two decades, largely been based on interconnecting single-processor compute nodes (PCs), and thereby offering computational power through parallel computing. Exploiting parallel computing has up to now been a niche technique for a small number of researchers, but this picture will soon have to change.

There are two clear hardware trends at the time of this writing. First, the architecture of the most powerful computers seems to be based on four ingredients: (1) interconnected, physically independent compute nodes, (2) a set of CPUs on each compute node, (3) a set of cores within each CPU, and (4) a set of specialized hardware accelerators such as general-purpose graphic processing units, known as GPUs. Second, the economically viable limit of CPU clock speed seems to have been approached, implying that further increase of the computational power in personal computers will rely on adding additional cores and GPUs, whose utilization requires parallel computing. The conclusion is therefore that the efficiency of serial computers has approached its limit, and increased hardware efficiency will be primarily based on parallel computing.

A good portion of the software environments mentioned in the previous section work in parallel, but the requirements on the end user differ widely. Some packages run trivially on a parallel computer, while others will need expert knowledge in parallelization from the user. Most of the developments of parallel computing techniques and software have been based on the cluster architecture with distributed memory, where MPI [167] is used for communication between a set of connected single-CPU compute nodes. MPI will most likely continue its domination at the compute-node level, while techniques and tools for communication at the multicore level and between cores and GPUs are still subject to intensive research. GPUs are normally utilized through programming tools like CUDA [4] or OpenCL [26]. Since memory is shared between the cores, OpenMP [98] can be used as an alternative to MPI on a given CPU. Determining an optimal communication pattern and implementing a parallel code in easily maintainable and reusable software remain ongoing challenges.

Consider the case of solving PDEs by the finite element method. There are two standard parallelization approaches to this type of problem. The first approach applies domain decomposition: the domain is divided into many subdomains, and the PDEs with approximate boundary conditions can be solved concurrently on all the subdomains. The approximate conditions at internal subdomain boundaries hopefully converge to the correct values using iteration strategies like the additive Schwarz method. The other approach consists in computing element matrices and vectors in parallel and assembling these into a global, distributed coefficient matrix and a right-hand side vector. The linear system can then be solved using libraries for parallel linear system solution. The distribution of element matrices and vectors as well as the parts of the global linear system can be based on a physical partitioning of the domain into subdomains or on non-physical data partitioning algorithms.

The challenge is to effectively map either of these two approaches onto new hardware architectures. For example, should a subdomain be distributed as one per compute node or one per core? The first case requires multicore parallelization of the subdomain computations, while the latter implies fine-grained parallelism and small subdomains. Such questions show that the parallelization itself has become significantly more complex, and since it will be needed for all computing devices in the future, effective software solutions must be found. The goal of these solutions is to bring models and computing power into the hands of the practitioners without requiring them to gain expert knowledge of modern parallel computing in heterogeneous hardware environments.

Development of tools that abstract this complexity from end users is still at an early stage. For example, PETSc now has the ability to manage communication between an MPI process and a single GPU with its own memory through the use of derived vector and matrix classes that can be selected by a user at run time [276]. A similar capability exists in Trilinos through the Kokkos package [38], and similar abstract tools are being developed by the hardware makers themselves, though typically only generating code for their own hardware [96,203]. The primary design goal of the PETSc implementation is to speed up sparse matrix vector operations during Krylov linear system solves, rather than fine-grained coordination of a finite element assembly process. Hedge [15] is, in contrast, an example of an effort to provide a high-level interface for explicit DG assembly on GPUs [237] using run-time code generation and a Python-based interface to CUDA (PyCUDA) [236]. But again, identifying appropriate algorithms for exploiting GPUs and implementing them in high-quality software remains a challenge.

## 8. Conclusions

Both routine and evolving water resources models have been documented and used to consider broadly computational aspects that bear upon the solution of these models. The objectives of this work were to give guidance for the selection of appropriate methods and to highlight evolving and emerging methods that have the promise to impact the modeling water resources systems. Based upon this work several conclusions are drawn:

1. Modeling of water resources systems is a vibrant field with classes of newly developed models prime candidates for interdisciplinary efforts aimed at numerical simulator development and validation.
2. Many of the challenges in modeling of water resources systems come from the desire for ever increasing resolution in space and time, often originating in the desire to resolve multiscale phenomena.
3. High-level algorithms will continue to play a central role in modeling of water resources systems with the desirable properties of error estimation, error control, and parallel processing efficiency being driving considerations.
4. While low-order fixed grid discretization methods are routine and widely used, several alternative and evolving methods hold promise for water resources systems, including discontinuous Galerkin methods, multiscale finite element methods, and Krylov deferred correction methods.
5. Each promising new class of discretization method has both areas of application in water resources modeling where important advancements could likely be made from immediate application as well as open issues that require further work to mature into new, important areas of application and to ensure efficiency.
6. Most mechanistic models of water resources systems require the solution of a system of nonlinear and linear algebraic equations. Many aspects of these solution methods are reasonably mature, and resources exist to aid the model developer in the application of appropriate, efficient methods. Challenges will continue to exist in developing methods that scale well on the parallel computing platforms of the future and care in general is needed to ensure choices are made that enable efficient scaling.
7. Significant effort is needed to develop an efficient, scalable simulator for a sophisticated water resource model. The emergence of computational environments, tool kits, solver packages, meshing software, and graphics rendering software can ease the development and application process and help ensure that any modeling effort leverages available resources for jobs that have already been done well, while enabling the focus on the critical, novel tasks that remain. For more routine applications, the development of a state-of-the-art simulator can be simplified by using these available resources in which much of the heavy lifting has already been done.
8. Computer hardware trends include increasing numbers of nodes, cores, and special processing units. In short, increases in computing power will come primarily from an increase in core counts and not from the speed at which individual cores operate. This trend will have profound implications for all involved in large-scale modeling of water resources systems, and the need to produce simulators that are scalable on the machines of tomorrow will pose a continuing challenge for the foreseeable future.

## Acknowledgments

The work of CTM was supported by Department of Energy Grant DE-SC0002163 and National Science Foundation Grant CDI-0941235. The work of CND was supported by National Science Foundation Grant DMS 0915223. The work of MWF and CEK was supported by the Civil Works and Military Engineering 6.1 research programs of the U.S. Army Corps of Engineers and the Coastal and Hydraulics Laboratory Internal Research Investment Program. Permission was granted by the Chief of Engineers to publish this information. The work of TYH was supported by AFOSR MURI Grant FA9550–09-1–0613, Department of Energy Grant DE-FG02–06ER25727, and National Science Foundation Grant DMS-0908546. The work of JH was supported by NSF Grants DMS-0811130 and CDI-0941235. The work of CTK was supported by ARO Grant W911NF-11–1-0367, US Army Corps of Engineers

Contract W912HZ-11-P-0385, and NSF Grant CDI-0941253. The work of HPL was supported by a Center of Excellence grant from the Research Council of Norway to the Center for Biomedical Computing at Simula Research Laboratory.

## References

[1] Advanced Visualization Systems. <http://www.avs.com>.
[2] COMSOL Multiphysics finite element programming environment. <http://www.comsol.com/>.
[3] Cubit geometry and mesh generation toolkit. <http://cubit.sandia.gov>.
[4] CUDA for programming of GPUs. <http://developer.nvidia.com/cuda-toolkit>.
[5] deal.II finite element programming environment. <http://www.dealii.org>.
[6] Diffpack programming environment. <http://www.diffpack.com>.
[7] DUNE: Distributed and unified numerics environment for PDEs. <http://www.dune-project.org/>.
[8] EnSight engineering visualization software. <http://www.ensight.com>.
[9] FiPy finite volume programming environment. <http://matforge.org/fipy/>.
[10] GetDP finite element programming environment. <http://www.geuz.org/getdp/>.
[11] Getfem++ C++ finite element library. <http://download.gna.org/getfem/html/homepage/>.
[12] Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. <http://geuz.org/gmsh>.
[13] gnuplot homepage. <http://www.gnuplot.info>.
[14] Hdf5 data model, library, and file format. <http://www.hdfgroup.org/HDF5>.
[15] Hedge: Hybrid and Easy Discontinuous Galerkin Environment. <http://mathema.tician.de/software/hedge>.
[16] IDL development environment. <https://www.exelisvis.com/language/en-us/productsservices/idl.aspx>.
[17] libmesh C++ finite element library. <http://libmesh.sourceforge.net/>.
[18] Maple software. <http://www.maplesoft.com/>.
[19] Mathematica software. <http://www.wolfram.com/mathematica/>.
[20] MATLAB software. <http://www.mathworks.se/products/matlab/index.html>.
[21] MayaVi: The mayavi data visualizer. <http://mayavi.sourceforge.net>.
[22] Mesh generation software. <http://www.robertschneiders.de/meshgeneration/software.html>.
[23] Netgen mesh generator. <http://sourceforge.net/apps/mediawiki/netgen-mesher>.
[24] Numerical Python software package. <http://numpy.scipy.org/>.
[25] Octave software. <http://www.gnu.org/software/octave/>.
[26] OpenCL standard for programming of heterogeneous systems. <http://www.khronos.org/opencl/>.
[27] Opendx: The open source software project based on IBM's visualization data explorer. <http://www.opendx.org>.
[28] OpenFOAM finite volume programming environment for CFD. <http://www.openfoam.com/>.
[29] ParaView — open source scientific visualization. <http://www.paraview.org>.
[30] ParaView Co-Processing library. <http://www.vtk.org/Wiki/CoProcessing>.
[31] Proteus: Computational methods and simulation toolkit. <http://proteus.usace.army.mil>.
[32] Python programming language. <http://www.python.org>.
[33] R programming language. <http://www.r-project.org>.
[34] Sage software. <http://sagemath.org/>.
[35] Scilab software. <http://www.scilab.org/>.
[36] Sundance finite element programming environment. <http://www.math.ttu.edu/~klong/Sundance/html/index.html>.
[37] TetGen a quality tetrahedral mesh generator and a 3d Delaunay triangulator. <http://tetgen.berlios.de>.
[38] Trilinos/Kokkos: Core Kernels Package. <http://trilinos.sandia.gov/packages/kokkos/documentation.html>.
[39] VisIt visualization tool. <http://wci.llnl.gov/codes/visit>.
[40] VTK — the visualization toolkit. <http://www.vtk.org>.
[41] XDMF: eXtensible Data Model and Format. <http://www.xdmf.org>.
[42] Aarnes JE. On the use of a mixed multiscale finite element method for greater flexibility and increased speed or improved accuracy in reservoir simulation. SIAM Multiscale Model Simulat 2004;2:421–39.
[43] Aarnes JE, Efendiev Y. Mixed multiscale finite element for stochastic porous media flows. SIAM J Sci Comput 2008;5:2319–39.
[44] Aarnes JE, Hauge V, Efendiev Y. Coarsening of three-dimensional structured and unstructured grids for subsurface flow. Adv Water Res 2007;30(11):2177–93.
[45] Aarnes JE, Krogstad S, Lie K-A. A hierarchical multiscale method for two-phase flow based upon mixed finite elements and nonuniform grids. SIAM Multiscale Model Simulat 2006;5:337–63.
[46] Aavatsmark I. An introduction to multipoint flux approximations for quadrilateral grids. Comput Geosci 2002;6:405–32.
[47] Aavatsmark I, Barkve T, ØBøe, Mannseth T. Discretization on unstructured grids for inhomogeneous, anisotropic media. Part I: Derivation of the methods. SIAM J Sci Comput 1998;19(5):1700–16.
[48] Aavatsmark I, Eigestad GT, Heimsund B-O, Nordbotten JM, ian E. A new finite-volume approach to efficient discretization on challenging grids. SPE J 2010;15(3):658–69.
[49] Adjerid S, Massey TC. Flexible Galerkin finite element methods. In: Proceedings of the second MIT conference on computational fluid and solid mechanics, vol. 2, Elsevier Science Ltd., Cambridge, MA, 2003, pp. 1848–1850. <http://books.google.com/books?id=ArhOoKshAwQC>.
[50] Aizinger V, Dawson CN. A discontinuous Galerkin method for two-dimensional flow and transport in shallow water. Adv Water Res 2002;25(1):67–84.
[51] Amestoy Patrick, Duff Iain, LExcellent Jean-Yves, Koster Jacko. Mumps: a general purpose distributed memory sparse solver. In: Srevik Tor, Manne Fredrik, Gebremedhin Assefaw, Moe Randi, editors. Applied parallel computing. New paradigms for HPC in industry and academia. Lecture notes in computer science, vol. 1947. Berlin/Heidelberg: Springer; 2001. p. 121–30. ISBN 978-3-540-41729-3. http://dx.doi.org/10.1007/3-540-70734-4_16.10.1007/3-540-70734-4_16.
[52] Arbogast T. Implementation of a locally conservative numerical subgrid upscaling scheme for two-phase darcy flow. Comput Geosci 2002;6:453–81.
[53] Arbogast T, Bryant SL. A two-scale numerical subgrid technique for waterflood simulations. SPE J 2002;7(4):446–56.
[54] Armijo L. Minimization of functions having Lipschitz-continuous first partial derivatives. Pacific J Math 1966;16:1–3.
[55] Arnold DN. An interior penalty finite-element method with discontinuous elements. SIAM J Numer Anal 1982;19(4):742–60.
[56] Arnold DN, Brezzi F, Cockburn B, Marini LD. Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM J Numer Anal 2002;39(5):1749–79.
[57] Ashcraft Cleve, Grimes Roger. SPOOLES: an object-oriented sparse matrix library. In: Michael Heroux et al., editor. Proceedings of the 9th SIAM conference on parallel processing for scientific computing. Philadelphia: SIAM; 1999. CD-ROM, ISBN 978-0898714357.
[58] Atkins HL, Shu CW. Quadrature-free implementation of the discontinuous Galerkin method for hyperbolic equations. AIAA J 1996;36:775–82.
[59] Aziz K, Settari A. Petroleum reservoir simulation. London: Applied Science Publ., Ltd.; 1979.
[60] Babuška I, Caloz G, Osborn E. Special finite element methods for a class of second order elliptic problems with rough coefficients. SIAM J Numer Anal 1994;31:945–81.
[61] Babuška I, Osborn E. Generalized finite element methods: their performance and their relation to mixed methods. SIAM J Numer Anal 1983;20:510–36.
[62] Baker GA. Finite-element methods for elliptic equations using non-conforming elements. Math Comput 1977;31(137):45–59.
[63] Bakhtyar R, Barry DA, Yeganeh-Bakhtiary A, Ghaheri A. Numerical simulation of surf-swash zone motions and turbulent flow. Adv Water Res 2009;32:250–63.
[64] Bakhtyar R, Barry DA, Yeganeh-Bakhtiary A, Li L, Parlange J-Y, Sander GC. Numerical simulation of two-phase flow for sediment transport in the inner-surf and swash zones. Adv Water Res 2010;33:277–90.
[65] Bakhtyar R, Yeganeh-Bakhtiary A, Barry DA, Ghaheri A. Two-phase hydrodynamic and sediment transport modeling of wave-generated sheet flow. Adv Water Res 2009;32:1267–83.
[66] Balay S, Brown J, Buschelman K, Eijkhout V, Gropp W, Kaushik D, et al. PETSc Users Manual Revision 3.2, Technical report. Mathematics and Computer Science Division, Argonne National Laboratory; September 2011.
[67] Balay Satish, Brown Jed, Buschelman Kris, Eijkhout Victor, Gropp William D., Kaushik Dinesh, Knepley Matthew G., Lois Curfman McInnes, Smith Barry F., Zhang Hong. PETSc users manual. Technical report ANL-95/11 - Revision 3.1, Argonne National Laboratory; 2010.
[68] Balay Satish, Brown Jed, Buschelman Kris, Gropp William D., Kaushik Dinesh, Knepley Matthew G., McInnes Lois Curfman, Smith Barry F., Zhang Hong. PETSc Web page; 2011. <http://www.mcs.anl.gov/petsc>.
[69] Balay Satish, Gropp William D, McInnes Lois Curfman, Smith Barry F. Efficient management of parallelism in object oriented numerical software libraries. In: Arge E, Bruaset AM, Langtangen HP, editors. Modern software tools in scientific computing. Birkhäuser Press; 1997. p. 163–202.
[70] Barry DA, Bajracharya K, Miller CT. Alternative split-operator approach for solving chemical reaction groundwater transport models. Adv Water Res 1996;19(5):261–75.
[71] Barry DA, Miller CT, Culligan PJ, Bajracharya K. Analysis of split operator methods for nonlinear and multispecies groundwater chemical transport models. Math Comput Simulat 1997;43:331–41.
[72] Barry DA, Miller CT, Culligan-Hensley PJ. Temporal discretization errors in non-iterative split-operator approaches to solving chemical reaction/groundwater transport models. J Contamin Hydrol 1996;22(1/2):1–17.
[73] Barry DA, Prommer H, Miller CT, Engesgaard PK, Brun A, Zheng C. Modelling the fate of oxidisable organic contaminants in groundwater. Adv Water Res 2002;25(8–12):945–83.
[74] Bastian P, Helmig R. Efficient fully-coupled solution techniques for two-phase flow in porous media – Parallel multigrid solution and large scale computations. Adv Water Res 1999;23(3):199–216.
[75] Battiato I, Tartakovsky DM, Tartakovsky AM, Scheibe TD. Hybrid simulations of reactive transport in porous and fractured media. Adv Water Res 2011;34:1140–50.
[76] Bause M, Knabner P. Computation of variably saturated subsurface flow by adaptive mixed hybrid finite element methods. Adv Water Res 2004;27:565–81.
[77] Bazilevs Y, Calo VM, Cottrel JA, Hughes TJR, Reali A, Scovazzi G. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Comput Methods Appl Mech Eng 2007;197:173–201.

[78] Bear J. Hydraulics of groundwater. New York: McGraw-Hill; 1979.
[79] Becker R, Rannacher R. An optimal control approach to a posteriori error estimation in finite element methods. Acta Numer 2002;10:1–102.
[80] Bell LSJ, Binning PJ. A split operator approach to reactive transport with the forward particle tracking Eulerian–Lagrangian localized adjoint method. Adv Water Res 2004;27:323–34.
[81] Bell N, Dalton S, Olson LN. Exposing fine-grained parallelism in algebraic multigrid methods. SIAM J Sci Comput; 2011 [in review].
[82] Bergamaschi L, Putti M. Mixed finite elements and Newton-type linearizations for the solution of Richards' equation. Int J Numer Methods Eng 1999;45:1025–46.
[83] Berger MJ, George L, LeVeque RJ, Mandli KT. The GeoClaw software for depth-averaged flows with adaptive refinement. Adv Water Res 2011;34(9):1195–206.
[84] Berkowitz B. Characterizing flow and transport in fractured geological media: A review. Adv Water Res 2002;25(8–12):861–84.
[85] Brenan KE, Campbell SL, Petzold LR. The numerical solution of initial value problems in differential-algebraic equations. Philadelphia, PA: Society for Industrial and Applied Mathematics; 1996.
[86] Brezzi F, Lipnikov K, Shashkov M, Simoncini V. A new discretization methodology for diffusion problems on generalized polyhedral meshes. Comput Methods Appl Mech Eng 2007;196(37–40):3682–92.
[87] Briggs WL, Henson VE, McCormick S. A multigrid tutorial. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics; 2000.
[88] Brown PN, Byrne GD, Hindmarsh AC. VODE: a variable coefficient ode solver. SIAM J Sci Statist Comput 1989;10:1038–51.
[89] Bryant SL, Thompson KE. Theory, modeling and experiment in reactive transport in porous media. Curr Opin Colloid Inter Sci 2001;6:217–22.
[90] Bu S, Huang JF, Boyer TH, Miller CT. An evaluation of solution algorithms and numerical approximation methods for modeling an ion exchange process. J Comput Phys 2010;229(13):4996–5010.
[91] Cahn John W, Hilliard John E. Free energy of a nonuniform system. I. Interfacial free energy. J Chem Phys 1958;28(2):258–67.
[92] Callister SJ, Wilkins MJ, Nicora CD, Williams KH, Banfield JF, Verberkmoes NC, et al. Analysis of biostimulated microbial communities from two field experiments reveals temporal and spatial differences in proteome profiles. Environ Sci Technol 2010;44(23):8897–903.
[93] Campbell JC, Hyman JM, Shashkov MJ. Mimetic finite difference operators for second-order tensors on unstructured grids. Comput Math Appl 2002;44:157–73.
[94] Camporese M, Paniconi C, Putti M, Orlandini S. Surface-subsurface flow modeling with path-based runoff routing, boundary condition-based coupling, and assimilation of multisource observation data. Water Resour Res 2010;46(W02512).
[95] Carrayrou J, Mose R, Behra P. Operator-splitting procedures for reactive transport and comparison of mass balance errors. J Contamin Hydrol 2004;68(3–4):239–68.
[96] Catanzaro B, Garland M, Keutzer K. Copperhead: compiling an embedded data parallel language. Technical report 124. Electrical Engineering and Computer Sciences, University of California at Berkeley; 2010.
[97] Celia MA, Bouloutas ET, Zarba RL. A general mass-conservative numerical solution for the unsaturated flow equation. Water Resour Res 1990;26(7):1483–96.
[98] Chapman B, Jost G, van der Pas R. Using OpenMP: portable shared memory parallel programming. The MIT Press; 2007.
[99] Chavent G, Jaffré J. Mathematical models and finite elements for reservoir simulation. Amsterdam: North-Holland; 1986.
[100] Chen Z, Hou TY. A mixed multiscale finite element method for elliptic problems with oscillating coefficients. Math Comput 2002;72:541–76.
[101] Cheng H-P, Howington SE, Farthing MW, McGrath CJ, Cheng J-RC..A generic reaction-based biogeochemical simulator (rbbgcs), version 1.0. Technical report, U.S.A.C.E Engineer Research and Development Center; 2010.
[102] Chow E, Cleary AJ, Falgout RD. Design of the hypre preconditioner library. In: Henderson M, Anderson C, Lyons S, editors. Proceedings of the SIAM workshop on object oriented methods for inter-operable scientific and engineering computing. SIAM; 1998.
[103] Chu CC, Graham I, Hou TY. The accuracy of multiscale finite element methods for high-contrast elliptic interface problems [in preparation].
[104] Cockburn B, Hou S, Shu CW. TVB Runge-Kutta local projection discontinuous Galerkin finite elemenet method for conservation laws IV: The multidimensional case. Math Comput 1990;54:545–81.
[105] Cockburn B, Karniadakis E, Shu CW. The development of discontinuous Galerkin methods. In: Cockburn B, Karniadakis E, Shu CW, editors. Lecture notes in computer science and engineering. New York: Springer-Verlag; 2000. p. 3–50.
[106] Cockburn B, Lin S, Shu CW. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation law III: One dimensional systems. J Comput Phys 1989;84:93–113.
[107] Cockburn B, Schötzau D, Wang J. Discontinuous Galerkin methods for incompressible elastic materials. Comput Methods Appl Mech Eng 2006;195:3184–204.
[108] Cockburn B, Shu C-W. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework. Math Comput 1989;52:411–35.
[109] Cockburn B, Shu CW. The Runge-Kutta local projection rho-1-discontinuous-Galerkin finite-element method for scalar conservation-laws. Rairo-Mathematical Modelling and Numerical Analysis-Modelisation Mathematique Et Analyse Numerique 1991;25(3):337–61.
[110] Cockburn B, Shu CW. The Runge-Kutta discontinuous Galerkin method for conservation laws V – multidimensional systems. J Comput Phys 1998;141(2):199–224.
[111] Cueto-Felgueroso L, Juanes R. A phase field model of unsaturated flow. Water Resour Res 2009;45:W10409. http://dx.doi.org/10.1029/2009WR007945.
[112] Davis TA. Algorithm 832: Umfpack, an unsymmetric-pattern multifrontal method. ACM Trans Math Soft 2004;30(2):196–9.
[113] Davis TA. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. ACM Trans Math Soft 2004;30(2):165–95.
[114] Davis TA. Direct methods for sparse linear systems. Number 2 in fundamentals of algorithms. Philadelphia: SIAM; 2006.
[115] Davis TA, Duff IS. An unsymmetric-pattern multifrontal method for sparse lu factorization. SIAM J Matrix Anal Appl 1997;38(1):140–58.
[116] Davis TA, Duff IS. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. ACM Trans Math Soft 1999;25(1):1–19.
[117] Dawson C, Kubatko EJ, Westerink J, Trahan C, Mirabito C, Michoski C, et al. Discontinuous Galerkin methods for modeling hurricane storm surge. Adv Water Res 2011;34(9):1165–76.
[118] Dawson C, Sun SY, Wheeler MF. Compatible algorithms for coupled flow and transport. Comput Methods Appl Mech Eng 2004;193(23–26):2565–80.
[119] Dawson CN, Aizinger V. A discontinuous Galerkin method for three-dimensional shallow water equations. J Sci Comput 2005;22(1):245–67.
[120] Dawson CN, Proft J. Coupling of continuous and discontinuous Galerkin methods for transport problems. Comput Methods Appl Mech Eng 2002;191(29–30):3213–31.
[121] Delshad M, Pope GA, Sepehrnoori K. A compositional simulator for modeling surfactant enhanced aquifer remediation, 1. Formulation. J Contamin Hydrol 1996;23(4):303–27.
[122] Demmel JW. Applied numerical linear algebra. Philadelphia: SIAM; 1997.
[123] Demmel James W, Eisenstat Stanley C, Gilbert John R, Li Xiaoye S, Liu Joseph WH. A supernodal approach to sparse partial pivoting. SIAM J Matrix Anal Appl 1999;20(3):720–55.
[124] Demmel James W, Gilbert John R, Li Xiaoye S. An asynchronous parallel supernodal algorithm for sparse gaussian elimination. SIAM J Matrix Anal Appl 1999;20(4):915–52.
[125] Dietrich JC, Trahan CJ, Howard MT, Fleming JG, Weaver RJ, Tanaka S, Yu L, Luettich Jr RA, Dawson CN, Westerink JJ, Wells G, Lu A, Vega K, Kubach A, Dresback KM, Kolar RL, Kaiser C, Twilley RR. Surface trajectories of oil transport along the northern coastline of the gulf of mexico. Continental Shelf Res ; 2012. ISSN 0278-4343. URL <http://www.sciencedirect.com/science/article/pii/S0278434312000799>.
[126] Dixon P, Brown D, Seitz R, Gorton I, Finsterle S, Moulton D, Steefel C, Freshley M, Hubbard S. ASCEM advanced simulation capability for environmental management; 2010. <http://ascemdoe.org>.
[127] Douglas J, Dupont T. Interior penalty procedures for elliptic and parabolic galerkin methods computing methods in applied sciences, vol. 58. Berlin/Heidelberg: Springer; 1976. Available from: <http://dx.doi.org/10.1007/BFb0120591>. p. 207–16.
[128] Durlofsky LJ. Numerical calculation of equivalent grid block permeability tensors for heterogeneous porous media. Water Resour Res 1991;27(5):699–708.
[129] Durlofsky LJ. Accuracy of mixed and control volume finite element approximations to Darcy velocity and related quantities. Water Resour Res 1994;30(4):965–73.
[130] Dutt A, Greengard L, Rokhlin V. Spectral deferred correction methods for ordinary differential equations. Bit Numer Math 2000;40(2):241–66.
[131] Edwards MG. M-matrix flux splitting for general full tensor discretization operators on structured and unstructuredgrids. J Comput Phys 2000;160:1–28.
[132] Edwards MG, Rogers CF. Finite volume discretization with imposed flux continuity for the general tensor pressure equation. Comput Geosci 1998;2:259–90.
[133] Efendiev Y, Galvis J, Thomines F. A systematic coarse-scale model reduction technique for parameter-dependent flows in highly heterogeneous media and its applications [submitted for publication].
[134] Efendiev Y, Galvis J, Wu XH. Multiscale finite element methods for high-contrast problems using local spectral basis functions. J Comput Phys 2011;230(4):937–55.
[135] Efendiev Y, Ginting V, Hou TY, Ewing R. Accurate multiscale finite element methods for two-phase flow simulations. J Comput Phys 2006;220(1):155–74.
[136] Efendiev Y, Hou T. Multiscale finite element methods. Theory and applications. Springer; 2009.
[137] Efendiev Y, Hou TY, Wu XH. Convergence of a nonconforming multiscale finite element method. SIAM J Numer Anal 2000;37:888–910.
[138] Eisenstat SC, Walker HF. Choosing the forcing terms in an inexact Newton method. SIAM J Sci Comput 1996;17(1):16–32.
[139] Enright D, Losasso F, Fedkiw R. A fast and accurate semi-lagrangian particle level set method. Comput Struct 2005;83:479–90.
[140] Ensig-Karup AP, Hesthaven JS, Bingham HB, Warburton T. DG-FEM solution for nonlinear wave-structure interaction using Boussinesq-type equations. Coastal Eng 2008;55:197–208.
[141] Epshteyn Y, Riviére B. Fully implicit discontinuous finite element methods for two-phase flow. Appl Numer Math 2007;57:383–401.

[142] Eriksson K, Johnson C. Adaptive finite elemtn methods for parabolic problems, 1. A linear model problem. SIAM J Numer Anal 1991;28(1):43–77.
[143] Eriksson K, Johnson C. Adaptive finite-element methods for parabolic problems. 5. Long-time integration. SIAM J Numer Anal 1995;32(6):1750–63.
[144] Ern A, Guermond J-L. Theory and practice of finite elements. Applied mathematical sciences, vol. 159. Springer; 2004.
[145] Ern A, Mozolevski I, Schuh L. Discontinuous Galerkin approximation of two-phase flows in heterogeneous porous media with discontinuous capillary pressures. Comput Methods Appl Mech Eng 2010;199:1491–501.
[146] Estep D. Multiscale methods: bridging the scales in science and engineering, chapter error estimates for multiscale operator decomposition for multiphysics models. Oxford University Press; 2009.
[147] Estep D, Ginting V, Ropp D, Shadid JN, Tavener S. A posteriori–a priori analysis of multiscale operator splitting. SIAM J Numer Anal 2008;46(3):1116–46.
[148] Estep D, Holst M, Mikulencak D. Accounting for stability: a posterior error estimates based on residual and variational analysis. Commun Numer Methods Eng 2002;18(1):15–30.
[149] Ethridge F, Greengard L. A new fast-multipole accelerated poisson solver in two dimensions. SIAM J Sci Comput 2001;23(3):741–60.
[150] Evans LC. Partial differential equations. Graduate studies in mathematics. Providence: AMS; 1998.
[151] Fabian N, Moreland K, Thompson D, Bauer A, Marion P, Geveci B, et al. The paraview coprocessing library: a scalable, general purpose *in situ* visualization library. In: LDAV 2011, IEEE symposium on large-scale data analysis and visualization. IEEE; 2011.
[152] Fang Y, Scheibe TD, Mahadevan R, Garg S, Long PE, Lovley DR. Direct coupling of a genome-scale microbial *in silico* model and a groundwater reactive transport model. J Contamin Hydrol 2011;122(1–4):96–103.
[153] Fang Y, Yeh G-T, Burgos W. A general paradigm to model reaction-based biogeochemical processes in batch systems. Water Resour Res 2003;39(4). http://dx.doi.org/10.1029/2002WR001694.
[154] Farthing MW, Miller CT. A comparison of high-resolution, finite-volume, adaptive-stencil schemes for simulating advective–dispersive transport. Adv Water Res 2000;24(1):29–48.
[155] Flemisch B, Darcis M, Erbertseder K, Faigle B, Lauser A, Mosthaf K, et al. DuMu$^x$: Dune for multi-phase, component, scale, physics, flow and transport in porous media. Adv Water Res 2011;34(9):1102–12.
[156] Freund RW. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. SIAM J Sci Comput 1993;14:470–82.
[157] Gasda SE, Farthing MW, Kees CE, Miller CT. Adaptive split-operator methods for modeling transport phenomena in porous medium systems. Adv Water Res 2011;34(10):1268–82.
[158] Geiser J. Consistency of iterative operator-splitting methods: theory and applications. Numer Methods Partial Diff Equat 2009;26:135–58.
[159] Govaerts WJF. Numerical methods for bifurcations of dynamic equlibria. Philadelphia: SIAM; 2000.
[160] Graham IG, Lechner P, Scheichl R. Domain decomposition for multiscale PDEs. Numer Math; 2007. http://dx.doi.org/10.1007/s00211-007-0074-1.
[161] Gray WG, Miller CT. Thermodynamically constrained averaging theory approach for modeling flow and transport phenomena in porous medium systems: 1. Motivation and overview. Adv Water Res 2005;28(2):161–80.
[162] Gray WG, Miller CT. Thermodynamically constrained averaging theory approach for modeling flow and transport phenomena in porous medium systems: 5. Single-fluid-phase transport. Adv Water Res 2009;32(5):681–711.
[163] Gray WG, Miller CT. Thermodynamically constrained averaging theory approach for modeling flow and transport phenomena in porous medium systems: 8. Interface and common curve dynamics. Adv Water Res 2010;33(12):1427–43.
[164] Gray WG, Miller CT. TCAT analysis of capillary pressure in non-equilibrium, two-fluid-phase, porous medium systems. Adv Water Res 2011;34(6):770–8.
[165] Greengard L, Gueyffier D, Martinsson PG, Rokhlin V. Fast direct solvers for integral equations in complex three-dimensional domains. Acta Numer 2009;18:243–75.
[166] Greengard L, Rokhlin V. A fast algorithm for particle simulations. J Comput Phys 1987;73(2):325–48.
[167] Gropp W, Lusk E, Skjellum A. Using MPI – portable parallel programming with the message-passing interface. 2nd ed. The MIT Press; 1999.
[168] Guillou A, Soule JL. La résolution numérique des problèmes différentiels aux conditions initiales par des méthodes de collocation. Revenue française d'informatique et de recherche opérationnelle 1969;3:17–44.
[169] Gulbransen AF, Hauge VL, Lie K-A. A multiscale mixed finite-element method for vuggy and naturally-fractured reservoirs. SPE J 2009.
[170] Guo BY, Wang ZQ. Legendre–Gauss collocation methods for ordinary differential equations. Adv Comput Math 2009;30(3):249–80.
[171] Gupta A, Karypis G, Kumar V. Highly scalable parallel algorithms for sparse matrix factorization. IEEE Trans Parallel Distribut Syst 1997;8:502–20.
[172] Hackbusch W. Multigrid methods and applications, computational mathematics. Springer-Verlag; 1985.
[173] Haga JB, Mardal K-A, Alnæs MS. cbc.block: flexible library for block preconditioning; 2011. <https://launchpad.net/cbc.block>.
[174] Haga JB, Osnes H, Langtangen HP. Efficient block preconditioners for the coupled equations of pressure and deformations in highly discontinuous media. Int J Anal Numer Methods Geomech 2011;35(13):1466–82. doi:10.1002/nag.973.
[175] Haggerty R, Gorelick SM. Multiple-rate mass transfer for modeling diffusion and surface reactions in media with pore-scale heterogeneity. Water Resour Res 1995;31(10):2383–400.
[176] Hairer E, Lubich C, Wanner G. Geometric numerical integration: structure-preserving algorithms for ordinary differential equations. Springer; 2006. <http://books.google.com/books?id=T1TaNRLmZv8C>.
[177] Hairer E, Norsett SP, Wanner G. Solving ordinary differential equations I, Nonstiff problems. Springer; 1987.
[178] Hairer E, Wanner G. Solving ordinary differential equations II, Stiff and differential–algebraic problems. Springer; 1991.
[179] Hajibeygi H, Bonfigli G, Hesse MA, Jenny P. Iterative multiscale finite-volume method. J Comput Phys 2008;227(19):8604–21.
[180] Hajibeygi H, Lunati I. Accurate and efficient simulation of multiphase flow in a heterogeneous reservoir by using error estimate and control in the multiscale finite-volume framework. In: SPE reservoir simulation symposium, 21–23 February, The Woodlands, Texas, USA; 2011.
[181] Hammond GE, Valocchi AJ, Lichtner P. Application of Jacobian-free NewtonKrylov with physics-based preconditioning to biogeochemical transport. Adv Water Res 2005;28(4):359–76.
[182] Helmig R. Multiphase flow and transport processes in the subsurface: a contribution to the modeling of hydrosystems. Berlin: Springer-Verlag; 1997.
[183] Heroux Michael A., Bartlett Roscoe A., Howle Vicki E., Hoekstra Robert J., Hu Jonathan J., Kolda Tamara G., Lehoucq Richard B., Long Kevin R., Pawlowski Roger P., Phipps Eric T., Salinger Andrew G., Thornquist Heidi K., Tuminaro Ray S., Willenbring James M., Williams Alan, Stanley Kendall S.. An overview of the Trilinos project. Technical report 3, Sandia National Laboratories; 2005.
[184] Herrera PA, Massabo M, Beckie R. A meshless method to simulate solute transport in heterogeneous porous media. Adv Water Res 2009;32:413–29.
[185] Hestenes MR, Steifel E. Methods of conjugate gradient for solving linear systems. J Res Nat Bureau Standards 1952;49:409–36.
[186] Hesthaven JS, Warburton T. Nodal discontinuous Galerkin methods: algorithms, analysis, and applications. New York: Springer; 2008. <http://books.google.com/books?id=APQkDOmwyksC>.
[187] Hesthaven JS, Warburton T. Nodal high-order methods on unstructured grids I. Time-domain solution of Maxwells equations. J Comput Phys 2002;181(186–221).
[188] Higham DJ, Higham NJ. MATLAB guide. Philadelphia: SIAM; 2000.
[189] Hirt CW, Nichols BD. Volume of fluid (vof) method for the dynamics of free boundaries. J Comput Phys 1981;39:201–25.
[190] Hoffman J, Johnson C. A new approach to turbulence modeling. Comput Meth Appl Mech Eng 2006;23-24:2865–80.
[191] Hoteit H, Firoozabadi A. Numerical modeling of two-phase flow in heterogeneous permeable media with different capillary pressures. Adv Water Res 2008;31(1):56–73.
[192] Hou TY, Wu XH, Zhang Y. Removing the cell resonance error in the multiscale finite element method via a Petrov–Galerkin formulation. Commun Math Sci 2004;2(2):185–205.
[193] Hou TY, Wu XH. A multiscale finite element method for elliptic problems in composite materials and porous media. J Comput Phys 1997;134:169–89.
[194] Hou TY, Wu XH. A multiscale finite element method for PDEs with oscillatory coefficients. In: Hackbusch W, Wittum G, editors. Proceedings of 13th GAMM-Seminar Kiel on numerical treatment of multi-scale problems, January 24–26, 1997. Notes on Numerical Fluid Mechanics, vol. 70. Vieweg-Verlag; 1999.
[195] Houston P, Schwab C, Suli E. Discontinuous *hp*-finite element methods for advection–diffusion–reaction problems. SIAM J Numer Anal 2002;39(6):2133–63.
[196] Huang J, Jia J, Minion M. Accelerating the convergence of spectral deferred correction methods. J Comput Phys 2006;214(2):633–56.
[197] Huang J, Jia J, Minion M. Arbitrary order Krylov deferred correction methods for differential algebraic equations. J Comput Phys 2007;221:739.
[198] Huber R, Helmig R. Multiphase flow in heterogeneous porous media: a classical finite element method versus an implicit pressure-explicit saturation-based mixed finite element-finite volume approach. Int J Numer Methods Fluids 1999;29:899–920.
[199] Huber R, Helmig R. Node-centered finite volume discretizations for the numerical simulation of multiphase flow in heterogeneous porous media. Comput Geosci 2000;4(2):141–64.
[200] Hunter J, Dale D, Droettboom M. Matplotlib: open source plotting package in Python; 2008. <http://matplotlib.sourceforge.net/>.
[201] Hutter K, Jöhnk KD. Continuum methods of physical modeling: continuum mechanics, dimensional analysis, turbulence. Springer; 2004.
[202] Imhoff PT, Farthing MW, Gleyzer SN, Miller CT. The evolving interface between clean and NAPL-contaminated regions in two-dimensional porous media. Water Resour Res 2002;38(6):1093.
[203] Intel parallel building blocks. Intel parallel building blocks. <http://software.intel.com/en-us/articles/intel-parallel-building-blocks>.
[204] ITAPS Geometry and Meshing Software. ITAPS geometry and meshing software. <http://www.itaps.org>.
[205] Jackson ABS, Miller CT, Gray WG. Thermodynamically constrained averaging theory approach for modeling flow and transport phenomena in porous medium systems: 6. Two-fluid-phase flow. Adv Water Res 2009;32(6):779–95.
[206] P. Janssen, The Interaction of Ocean Waves and Wind, Cambridge; 2004.
[207] Jenkins EW, Berger RC, Hallberg JP, Howington SE, Kelley CT, Schmidt JH, et al. A two-level aggregation-based Newton–Krylov–Schwarz method for

hydrology. In: Keyes DE, Ecer A, Periaux J, Satofuka N, editors. Parallel computational fluid dynamics 1999. North Holland; 2000. p. 257–64.
[208] Jenkins EW, Kees CE, Kelley CT, Miller CT. An aggregation-based domain decomposition preconditioner for groundwater flow. SIAM J Sci Comput 2001;23(2):430–41.
[209] Jenny P, Lee SH, Tchelepi HA. Adaptive fully implicit multi-scale finite-volume method for multi-phase flow and transport in heterogeneous porous media. J Comput Phys 2006;217:627–41.
[210] Jenny P, Lee SH, Tchelepi H. Multi-scale finite volume method for elliptic problems in subsurface flow simulation. J Comput Phys 2003;187:47–67.
[211] Jia J, Huang J. Krylov deferred correction accelerated method of lines transpose for parabolic problems. J Comput Phys 2008;227(3):1739–53.
[212] Jones E, Oliphant T, Peterson P, et al. SciPy: Open source scientific tools for Python; 2001. <http://www.scipy.org..
[213] Jones JE, Woodward CS. Newton–krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems. Adv Water Res 2001;24(7):763–74.
[214] Kanney JF, Miller CT, Barry DA. Comparison of approaches for approximating nonlinear transport and reaction problems. Adv Water Res 2003;26(4):353–72.
[215] Kanschat G. Preconditioning methods for local discontinuous Galerkin discretizations. SIAM J Sci Comput 2003;25(3):815–31.
[216] Kavetski D, Binning P, Sloan SW. Adaptive time stepping and error control in a mass conservative numerical solution of the mixed form of Richards' equation. Adv Water Res 2001;24(6):595–605.
[217] Kees CE, Farthing MW, Howington SE, Jenkins EW, Kelley CT. Nonlinear multilevel iterative methods for multiscale models of air/water flow in porous media. In: Proceedings of computational methods in water resources XVI; 2006.
[218] Kees CE, Miller CT. Higher order time integration methods for two-phase flow. Adv Water Res 2002;25(2):159–77.
[219] Kees CE, Miller CT, Jenkins EW, Kelley CT. Versatile two-level Schwarz preconditioners for multiphase flow. Comput Geosci 2003;7(2):91–114.
[220] Kees CE, Akkerman I, Farthing MW, Bazilevs Y. A conservative level set method suitable for variable-order approximations and unstructured meshes. J Comput Phys 2011;230(11):4536–58.
[221] Kees CE, Farthing MW. Parallel computational methods and simulation for coastal and hydraulic applications using the proteus toolkit. In: SC11:PyHPC workshop; 2011.
[222] Kees CE, Farthing MW, Dawson CN. Locally conservative, stabilized finite element methods for variably saturated flow. Comput Methods Appl Mech Eng 2008;197:4610–25.
[223] Keller HB. Lectures on numerical methods in bifurcation theory. Tata Institute of Fundamental Research. Lectures on mathematics and physics. New York: Springer-Verlag; 1987.
[224] Kelley CT. Iterative methods for linear and nonlinear equations. Vol. 16 in frontiers in applied mathematics. Philadelphia: SIAM; 1995.
[225] Kelley CT. Solving nonlinear equations with Newton's method. Vol. 1 in fundamentals of algorithms. Philadelphia: SIAM; 2003.
[226] Kelley CT, Keyes DE. Convergence analysis of pseudo-transient continuation. SIAM J Numer Anal 1998;35:508–23.
[227] Ketcheson DI, Mandli KT. PyClaw software. <http://github.com/clawpack/pyclaw>.
[228] Ketcheson DI, Mandli KT, Ahmadia A, Alghamdi A, Quezada M, Parsani M, et al. Accessible, extensible, scalable tools for wave propagation problems. SIAM J Sci Comput. [submitted for publication].
[229] Kippe V, Aarnes JE, Lie K-A. A comparison of multiscale methods for elliptic problems in porous media flow. Comput Geosci 2008;12:377–98.
[230] Kirby RC, Knepley MG, Logg A, Scott LR. Optimizing the evaluation of finite element matrices. SIAM J Sci Comput 2005;27(6):741–58. ISSN 1064-8275.
[231] Kirby RC, Logg A. A compiler for variational forms. ACM Trans Math Softw 2006;32:417–44. <http://dx.doi.org/10.1145/1163641.1163644>.
[232] Kirby RC, Logg A. Efficient compilation of a class of variational forms. ACM Trans Math Soft 2007;33(3).
[233] Kirby RC, Logg A. Benchmarking domain-specific compiler optimizations for variational forms. ACM Trans Math Soft 2008;35(2):1–18. <http://dx.doi.org/10.1145/1377612.1377614>.
[234] Kirkland MR, Hills RG, Wierenga PJ. Algorithms for solving Richards' equation for variably saturated soils. Water Resour Res 1992;28(8):2049–58.
[235] Klausen RA, Russell TF. Relationships among some locally conservative discretization methods which handle discontinuous coefficients. Comput Geosci 2004;8(4):341–77.
[236] Klöckner A, Pinto N, Lee Y, Catanzaro B, Ivanov P, Fasih A. PyCUDA and PyOpenCL: a scripting-based approach to GPU run-time code generation. Parallel Comput, in press. http://dx.doi.org/10.1016/j.parco.2011.09.001.
[237] Klöckner A, Warburton T, Bridge J, Hesthaven JS. Nodal discontinuous Galerkin methods on graphics processors. J Comput Phys 2009;228(21):7863–82.
[238] Knabner P, Tapp C, Thiele K. Adaptivity in the finite volume discretization of variable density flows in porous media. Phys Chem Earth Part B – Hydrol Oceans Atmos 2001;26(4):319–24.
[239] Knauss KG, Johnson JW, Steefel CI. Evaluation of the impact of $CO_2$, co-contaminant gas, aqueous fluid and reservoir rock interactions on the geologic sequestration of $CO_2$. Chem Geol 2005;217:339–50.
[240] Knoll DA, Keyes DE. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. J Comput Phys 2004;193(2):357–97.
[241] Kolditz O. Opengeosys; 2009. <http://www.ufz.de/index.php?en=18345>.
[242] Komen GJ, Cavaleri L, Donelan M, Hasselmann K, Hasselman S, Janssen P. Dynamics and modeling of ocean waves, Cambridge; 1994.
[243] Kräutle S, Knabner P. A new numerical reduction scheme for fully coupled multicomponent transport-reaction problems in porous media. Water Resour Res 2005;41. http://dx.doi.org/10.1029/2004WR003624.
[244] Kräutle S, Knabner P. A reduction scheme for coupled multicomponent transport-reaction problems in porous media: generalization to problems with heterogeneous equilibrium reactions. Water Resour Res 2007;43. http://dx.doi.org/10.1029/2005WR004465.
[245] Kubatko EJ, Bunya S, Dawson C, Westerink JJ, Mirabito C. A performance comparison of continuous and discontinuous finite element shallow water models. J Sci Comput 2009;40(1–3):315–39.
[246] Kubatko EJ, Bunya S, Dawson C, Westerink J. Dynamic *p*-adaptive Runge-Kutta discontinuous Galerkin methods for the shallow water equations. Comput Methods Appl Mech Eng 2009;198(21–26):1766–74.
[247] Lake LW. Enhanced oil recovery. Englewood Cliffs, NJ: Prentice Hall; 1989.
[248] Langtangen HP. Computational partial differential equations – numerical methods and diffpack programming. Texts in computational science and engineering. Springer; 2003.
[249] Langtangen HP. Python scripting for computational science. Texts in computational science and engineering. Springer; 2008.
[250] Lee J, Greengard L. A fast adaptive numerical method for stiff two-point boundary value problems. SIAM J Sci Comput 1997;18(2):403–29.
[251] Lee SH, Wolfsteiner C, Tchelepi H, Jenny P, Lunati I. Method, apparatus and system for reservoir simulation using a multi-scale finite volume method including black oil modeling. Patent number: 7765091.
[252] LeSaint P, Raviart PA. On a finite element method for solving the neutron transport equation. Math Aspects Finite Elem Methods Partial Differ Equat 1974:89–145.
[253] Li H, Farthing MW, Dawson CN, Miller CT. Local discontinuous Galerkin approximations to Richards' equation. Adv Water Res 2007;30:555–75.
[254] Li H, Farthing MW, Miller CT. Adaptive local discontinuous Galerkin approximation to Richards' equation. Adv Water Res 2007;30:1883–901.
[255] Li Xiaoye S, Demmel James W. SuperLU_DIST: a scalable distributed-memory sparse direct solver for unsymmetric linear systems. ACM Trans Math Software 2003;29(2):110–40.
[256] Lichtner PC, Steefel CI, Oelkers EH. Reactive transport in porous media: general principles and applications to geochemical processes. Washington, DC: Mineralogical Society of America; 1996.
[257] Lighthill MJ, Whitham GB. On kinematic waves. I. Flood movement in long rivers. Proc Royal Soc London. Ser A, Math Phys Sci 1955;229(1178):281–316.
[258] Lin H-CJ, Richards DR, Yeh G-T, Cheng J-R, Cheng H-P. FEMWATER: a three-dimensional finite element computer model for simulating density-dependent flow and transport in variably saturated media. Technical report, U.S. Army Corps of Engineers Waterways Experiment Station, Vicksburg, MS; 1997.
[259] Lin P, Liu PL-F. A numerical study of breaking waves in the surf zone. J Fluid Mech 1998;359:239–64.
[260] Liu Y, MacFadyen A, Ji Z-G, Weisberg RH, editors. Monitoring and modeling the deepwater horizon oil spill: a record-breaking enterprise. Geophysical monograph series, vol. 195 AGU; 2011.
[261] Logg A, Wells GN. DOLFIN: automated finite element computing. ACM Trans Math Soft 2010;32(2):1–28. <http://dx.doi.org/10.1145/1731022.1731030>.
[262] Logg A, Wells GN, Hake, JE, et al. FEniCS finite element programming environment; 2003. <http://www.fenicsproject.org>.
[263] Lunati I, Jenny P. Multi-scale finite-volume method for three-phase flow influenced by gravity. In Proceedings of computational methods in water resources XVI; 2006.
[264] Lunati I, Jenny P. Multiscale finite-volume method for compressible multiphase flow in porous media. J Comput Phys 2006;216:616–36.
[265] Lunati I, Jenny P. Multiscale finite-volume method for density-driven flow in porous media. Comput Geosci 2008;12(3):337–50.
[266] Maier RS, Petzold LR. User's guide to DASKMP and DASPKF90. Technical report AHPCRC Preprint 93-034, Army High Performance Computing Research Center, Minneapolis, MN; 1993.
[267] Manteuffel Thomas A, Parter Seymour. Preconditioning and boundary conditions. SIAM J Numer Anal 1990;27:656–94.
[268] Mao X, Prommer H, Barry D, Langevin C, Panteleit B, Li L. Three-dimensional model for multi-component reactive transport with variable density groundwater flow. Environ Modell Software 2006;21(5):615–28.
[269] Maryška J, Severýn O, Vohralík M. Numerical simulation of fracture flow with a mixed-hybrid FEM stochastic discrete fracture network model. Comput Geosci 2004;8:217–34.
[270] Miller CT, Abhishek C, Farthing MW. A spatially and temporally adaptive solution of Richards' equation. Adv Water Res 2006;29:525–45.
[271] Miller CT, Abhishek C, Sallerson AB, Prins JF, Farthing MW. A comparison of computational and algorithmic advances for solving Richards' equation. In: C.T. Miller, M.W. Farthing, W.G. Gray, G.F. Pinder, editors. Proceedings of the XVth international conference on computational methods in water resources. Chapel Hill, North Carolina: Elsevier; 2004. p. 1131–45.
[272] Miller CT, Christakos G, Imhoff PT, McBride JF, Pedit JA, Trangenstein JA. Multiphase flow and transport modeling in heterogeneous porous media: challenges and approaches. Adv Water Res 1998;21(2):77–120.
[273] Miller CT, Gleyzer SN, Imhoff PT. Numerical modeling of NAPL dissolution fingering in porous media. In: Selim HM, Ma L, editors. Physical

nonequilibrium in soils: modeling and application. Ann Arbor, MI: Ann Arbor Press; 1998.
[274] Miller CT, Gray WG. Thermodynamically constrained averaging theory approach for modeling flow and transport phenomena in porous medium systems: 2. Foundation. Adv Water Res 2005;28(2):181–202.
[275] Miller CT, Rabideau AJ. Development of split-operator, Petrov–Galerkin methods to simulate transport and diffusion problems. Water Resour Res 1993;29(7):2227–40.
[276] Minden V, Smith B, Knepley MG. Preliminary implementation of PETSc using GPUs. In: Proceedings of the 2010 international workshop of GPU solutions to multiscale problems in science and engineering. Springer; 2011.
[277] Mirabito C, Dawson C, Kubatko EJ, Westerink JJ, Bunya S. Implementation of a discontinuous Galerkin morphological model on two-dimensional unstructured meshes. Comput Methods Appl Mech Eng 2011;200:1207–389.
[278] Mortensen M, Langtangen HP, Wells GN. A FEniCS-based programming framework for modeling turbulent flow by the Reynolds-averaged Navier–Stokes equations. Adv Water Res 2011;34(9):1082–101.
[279] Mortensen M, Langtangen HP, Wells GN. A FEniCS-based programming framework for modeling turbulent flow by the Reynolds-averaged Navier–Stokes equations. Adv Water Res 2011. http://dx.doi.org/10.1016/j.advwatres.2011.02.013.
[280] Mosé R, Siegel P, Ackerer P. Application of the mixed hybrid finite element approximation in a groundwater flow model: luxury or necessity? Water Resour Res 1994;30(11):3001–12.
[281] Mustapha H, Dimitrakopoulos R, Graf T, Firoozabadi A. An efficient method for discretizing 3d fractured media for subsurface flow and transport simulations. Int J Numer Methods Fluids 2011;67:651–70.
[282] Nagrath S, Jansen KE, Lahey RT. Computation of incompressible bubble dynamics with a stabilized finite element level set method. Comput Methods Appl Mech Eng 2005;194:4565–87.
[283] NetCDF Data Format. NetCDF data format. <http://www.unidata.ucar.edu/software/netcdf>.
[284] Nordbotten JM, Eigestad GT. Discretization on quadrilateral grids with improved monotonicity properties. J Comput Phys 2005;203(2):744–60.
[285] Olsson E, Kreiss G. A conservative level set method for two phase flow. J Comput Phys 2005;210:225–46.
[286] Oostrom M, Lenhard RJ. Comparison of relative permeability-saturation-pressure parametric models for infiltration and redistribution of a light nonaqueous-phase liquid in sandy porous media. Adv Water Res 1998;21(2):145–57.
[287] Osher S, Fedkiw R. Level set methods and dynamic implicit surfaces. Applied mathematical sciences, vol. 153. New York: Springer-Verlag; 2003.
[288] Owhadi H, Zhang L. Metric based up-scaling. Commun Pure Appl Math 2007;LX:675–723.
[289] Paniconi C, Putti M. A comparison of Picard and Newton iteration in the numerical solution of multidimensional variably saturated flow problems. Water Resour Res 1994;30(12):3357–74.
[290] Pérez Fernando, Granger Brian E. IPython: a system for interactive scientific computing. Comput Sci Eng 9 (3) (2007) 21–29, May 2007. <http://ipython.org>.
[291] Petzold LR. A description of DASSL: a differential/algebraic system solver. In: Stepleman RS et al., editors. Scientific computing. Amsterdam: North Holland; 1983. p. 65–8.
[292] Pini R, Krevor SCR, Benson SM. Capillary pressure and heterogeneity for the $CO_2$/water system in sandstone rocks at reservoir conditions. Adv Water Reour 2012;38:48–59.
[293] Ponce VM, Li R-M, Simons DB. Applicability of kinematic and diffusion models. J Hydraulics Division 1978;104(3):353–60.
[294] Press WH, Flannery BP, Teukolsky SA, Vattering WT. Numerical recipes: the art of scientific computing. London: Cambridge University Press; 1992.
[295] Prill F, Lukacova-Medvidova M, Hartmann R. Smoothed aggregation multigrid for the discontinuous Galerkin method. SIAM J Sci Comput 2009;31(5):3503–28.
[296] Prommer H, Barry D, Zheng C. MODFLOW/MT3DMS-based reactive multicomponent transport modeling. Ground Water 2003;41(2):247–57.
[297] Pruess K. TOUGH2: a general purpose numerical simulator for multiphase fluid and heat flow. Technical Report LBL-29400. Lawrence Berkeley Laboratory, Berkeley, CA; 1991.
[298] Puckett EG, Almgren AS, Bell JB, Marcus DL, Rider WJ. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. J Comput Phys 1997;130:269–82.
[299] Qu Y, Duffy CJ. A semidiscrete finite volume formulation for multiprocess watershed simulation. Water Resour Res 2007;43:W08419.
[300] Helmig R, Weiss A, Wohlmuth BI. Variational inequalities for modeling flow in heterogeneous porous media with entry pressure. Comput Geosci 2009;13:373–89.
[301] Rannacher R. Adaptive Galerkin finite element methods for partial differential equations. J Comput Appl Math 2001;128:205–33.
[302] Reed WH, Hill TR. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Laboratory; 1973.
[303] Riaz A, Hesse M, Tchelepi HA, Orr FM. Onset of convection in a gravitationally unstable diffusive boundary layer in porous media. J Fluid Mech 2006;548:87–111.
[304] Rider WJ, Kothe DB. Reconstructing volume tracking. J Comput Phys 1998;141:112–52.
[305] Rivière B. Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation. SIAM, Society for Industrial and Applied Mathematics; 2008. <http://books.google.com/books?id=JklTMrPZT3gC>.
[306] Rognes ME, Logg A. Exploring automated adaptivity and error control. In T.E. Simos, G. Psihoyios, C. Tsitouras, editors. AIP conference proceedings, vol. 1281; 2010. p. 794–7.
[307] Rotkin Vladimir, Toledo Sivan. The design and implementation of a new out-of-core sparse Cholesky factorization method. ACM Trans Math Soft 2004;30(1):19–46.
[308] Rupert CP, Miller CT. An analysis of polynomial chaos approximations for modeling single-fluid-phase flow in porous medium systems. J Comput Phys 2007;226(2):2175–205.
[309] Russell TF, Celia MA. An overview of research on Eulerian-Lagrangian localized adjoint methods (ELLAM). Adv Water Res 2002;25(8–12):1215–31.
[310] Saad Y, Schultz MH. GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 1986;7:856–69.
[311] Saadatpoor E, Bryant SL, Sepehrnoori K. New trapping mechanism in carbon sequestration. Transport Porous Med 2009;82:3–17.
[312] Saaltink MW, Ayora C, Carrera J. A mathematical formulation for reactive transport that eliminates mineral concentrations. Water Resour Res 1998;34(7):1649–56.
[313] Saaltink MW, Carrera J, Ayora C. On the behavior of approaches to simulate reactive transport. J Contamin Hydrol 2001;48:213–35.
[314] Sala M, Stanley K, Heroux M. Amesos: a set of general interfaces to sparse direct solver libraries. In: Proceedings of PARA'06 conference, Umea, Sweden; 2006.
[315] Sala M, Stanley K, Heroux M. On the design of interfaces to sparse direct solvers 2008;34(2).
[316] Salamon P, Fernández-Garcia D, Gómez-Hernández JJ. Modeling mass transfer processes using random walk particle tracking. Water Resour Res; 2006. p. W11417. http://dx.doi.org/10.1029/2006WR004927.
[317] Salinger Andrew G., Bou-Rabee Nawaf M., Pawlowski Roger P., Wilkes Edward D., Burroughs Elizabeth A., Lehoucq Richard B., Romero Louis A. LOCA 1.0 library of continuation algorithms: theory and implementation manual. Technical report SAND2002-0396, Sandia National Laboratory; March 2002.
[318] Scardovelli R, Zaleski S. Direct numerical simulation of free-surface and interfacial flow. Annual Rev Fluid Mech 1999;31:567–603.
[319] Schenk O, Gärtner K. Solving unsymmetric sparse systems of linear equations with pardiso. J Future Generat Comput Syst 2004;20(3):475–87.
[320] Schenk O, Gärtner K. On fast factorization pivoting methods for symmetric indefinite systems. Elec Trans Numer Anal 2006;23:158–79.
[321] Schubert J, Sanders BF, Smith MJ, Wrigt NG. Unstructured mesh generation and landcover-based resistance for hydrodynamic modeling of urban flooding. Adv Water Res 2008;31:1603–21.
[322] Sethian JA, Smereka P. Level set methods for fluid interfaces. Annual Rev Fluid Mech 2003;35:341–72.
[323] Shewchuk JR. Triangle: engineering a 2d quality mesh generator and Delaunay triangulator. In: Lin MC, Manocha D, editors. Applied computational geometry: towards geometric engineering. Lecture notes in computer science, vol. 1148. Springer-Verlag; 1996. p. 203–22.
[324] Siek J, Lumsdaine A. A modern framework for high-performance numerical linear algebra. In: Langtangen HP, Bruaset AM, Quak E, editors. Advances in software tools for scientific computing. Springer; 2000.
[325] Simpson MJ, Landman KA. Analysis of split operator methods applied to reactive transport with Monod kinetics. Adv Water Res 2007;30(9):2026–33.
[326] Simunek J, Vogel T, Genuchten MT. The SWMS_2D code for simulating water flow and solute transport in two-dimensional variably saturated media. Technical report, U.S. Salinity Laboratory, Agricultural Research Service, U.S. Department of Agriculture; 1994.
[327] Smith B, Bjørstad P, Gropp W. Domain decomposition: parallel multilevel methods for elliptic partial differential equations. Cambridge: Cambridge University Press; 1996.
[328] Solin P, Kuraz M. Solving the nonstationary Richards equation with adaptive hp-fem. Adv Water Res 2011;34(9):1062–81.
[329] Steefel CI, Macuarrie KTB. Approaches to modeling of reactive transport in porous media. In: Lichtner PC, Steefel CI, Oelkers EH, editors. Reactive transport in porous media: general principles and applications to geochemical processes. Washington, DC: Mineralogical Society of America; 1996. p. 83–129.
[330] Strang G. On the construction and comparison of difference schemes. SIAM J Numer Anal 1968;5(3):506–17.
[331] Stumm W, Morgan JJ. Aquatic chemistry. New York: John Wiley and Sons; 1996.
[332] Sussman M. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. J Comput Phys 2003;187:110–36.
[333] Sussman M, Smereka P, Osher S. A level set approach for computing solutions to incompressible two-phase flow. J Comput Phys 1994;114:146–59.
[334] Sussman M, Smereka P, Osher SJ. A level set approach for computing solutions to incompressible two-phase flows. J Comput Phys 1994;114:146–59.
[335] Tartakovsky AM, Tartakovsky DM, Scheibe TD, Meakin P. Hybrid simulations of reaction–diffusion systems in porous media. SIAM J Sci Comput 2008;30(6):2799–816.

[336] Tezduyar TE. Interface-tracking and interface-capturing techniques for finite element computation of moving boundaries and interfaces. Comput Methods Appl Mech Eng 2006;195:2983–3000.
[337] Tocci MD, Kelley CT, Miller CT. Accurate and economical solution of the pressure-head form of Richards' equation by the method of lines. Adv Water Res 1997;20(1):1–14.
[338] Tornberg A, Enquist B. A finite element based level-set method for multiphase flow applications. Comput Visualiz Sci 2000;3:93–101.
[339] Trangenstein JA. Multi-scale iterative techniques and adaptive mesh refinement for flow in porous media. Adv Water Res 2002;25(8–12):1175–213.
[340] Unverdi SO, Tryggvason G. A front-tracking method for viscous, incompressible, multi-fluid flows. J Comput Phys 1992;100:25–37.
[341] Valocchi AJ, Malmstead M. Accuracy of operator splitting for advection–dispersion–reaction problems. Water Resour Res 1992;28(5):1471–6.
[342] van der Vorst HA. Bi-CGSTAB: a fast and smoothly converging variant to Bi-CG for the solution of nonsymmetric systems. SIAM J Sci Statist Comput 1992;13:631–44.
[343] Vreugdenhil CB. Numerical methods for shallow-water flow. Kluwer Academic Publishers; 1992.
[344] Walker HF, Ni P. Anderson acceleration for fixed-point iterations. SIAM J Numer Anal 2011;49(4):1715–35.
[345] Watson IA, Crouch RS, Bastian P, Oswald SE. Advantages of using adaptive remeshing and parallel processing for modelling biodegradation in groundwater. Adv Water Res 2005;28:1143–58.
[346] Watson LT, Billups SC, Morgan AP. Algorithm 652: HOMPACK: a suite of codes for globally convergent homotopy algorithms. ACM Trans Math Software 1987;13:281–310.
[347] Watson SJ, Barry DA, Schotting RJ, Hassanizadeh SM. On the validity of Darcy's law for stable high-concentration displacements in granular porous media. Transport Porous Med 2002;47(2):149–67.
[348] Watson SJ, Barry DA, Schotting RJ, Hassanizadeh SM. Validation of classical density-dependent solute transport theory for stable, high-concentration-gradient brine displacements in coarse and medium sands. Adv Water Res 2002;25(6):611–35.
[349] Wheeler MF. Elliptic collocation-finite element method with interior penalties. SIAM J Numer Anal 1978;15(1):152–61.
[350] Wheeler MF, Peszynska M. Computational engineering and science methodologies for modeling and simulation of subsurface applications. Adv Water Res 2002;25(8-12):1147–73.
[351] Wheeler MF, Yotov I. A multipoint flux mixed finite element method. SIAM J Numer Anal 2006;44(5):2082–106.
[352] White MD, Oostrom M, Lenhard RJ. Modeling fluid flow and transport in variably saturated porous media with the STOMP simulator. 1. Nonvolatile three-phase model description. Adv Water Res 1995;18(6):353–64.
[353] Williams GA, Miller CT. An evaluation of temporally adaptive transformation approaches for solving Richards' equation. Adv Water Res 1999;22(8):831–40.
[354] Williams GA, Miller CT, Kelley CT. Transformation approaches for simulating flow in variably saturated porous media. Water Resour Res 2000;36(4):923–34.
[355] Wright K. Some relationships between implicit Runge-Kutta, collocation and Lanczos $\tau$ methods, and their stability properties. Bit Numer Math 1970;10(2):217–27.
[356] Wu XH, Efendiev Y, Hou TY. Analysis of upscaling absolute permeability. Discrete Continuous Dynam Syst Ser B 2002;2:185–204.
[357] Xu T, Sonnenthal E, Spycher N, Preuss K. TOUGHREACT user's guide: a simulation program for non-isothermal multiphase reactive geochemical transport in variably saturated geologic media. Technical report, Lawrence Berkeley National Laboratory; 2008.
[358] Zeidler E. Nonlinear functional analysis and its applications. II/B. Springer-Verlag; 1990. Nonlinear monotone operators, Translated from the German by the author and Leo F. Boron.
[359] Zhang B, Huang J. Order and stiffness of the Gauss Runge-Kutta methods for initial-boundary value problems. SIAM J Numer Anal [in review].
[360] Zhang D. Stochastic methods for flow in porous media: coping with uncertainties. San Diego, CA: Academic Press; 2002.
[361] Zyvoloski GA, Robinson BA, Dash ZV, Trease LL. Summary of the models and methods for the FEHM application—a finite-element heat- and mass-transfer code. Technical Report LA-13307-MS, Los Alamos National Laboratory, Los Alamos, NM; 1997.