The SRFR 5 Modeling System for Surface Irrigation

E. Bautista, A.M.ASCE[1]; J. L. Schlegel[2]; and A. J. Clemmens, M.ASCE[3]

Abstract: The *SRFR* program is a modeling system for surface irrigation. It is a central component of *WinSRFR*, a software package for the hydraulic analysis of surface irrigation systems. *SRFR* solves simplified versions of the equations of unsteady open channel flow coupled to a user-selected infiltration model. *SRFR* was reprogrammed using a modern object-oriented architecture with the objective of facilitating its continued development and, thus, the addition of new modeling options and functionalities. The upgraded software is *SRFR 5*. An important component of the modeling system is the code that manages computational incidents. While the computational methods used by *SRFR 5* have been widely tested and are generally robust, calculations are prone to failures due to problems with the solution of the nonlinear finite difference system of equations, discretization problems, and problems with the determination of the appropriate boundary condition at each computational time step. Code that manages computational incidents has been greatly enhanced in *SRFR 5*. *SRFR 5* has undergone systematic testing, partly to fix programming errors, but also to improve computations leading to unacceptable volume balance errors. This article discusses the *SRFR 5* modeling system and provides details about its computational methods, architecture, and new features. **DOI: 10.1061/(ASCE)IR.1943-4774.0000938.** © *2015 American Society of Civil Engineers.*

Introduction

WinSRFR (Bautista et al. 2009) is a software package for the hydraulic analysis of surface irrigation systems. A central component of the software is the unsteady flow modeling system *SRFR*. *WinSRFR* includes tools for irrigation system evaluation, design, and operational analysis, all of which are supported by *SRFR* calculations.

SRFR predicts the coupled surface and subsurface flow of water in a surface irrigation system as a function of system type, geometry, hydraulic resistance and infiltration characteristics of the soil, and the rate of water inflow as a function of time. These options have been previously described (e.g., Bautista et al. 2009; Strelkoff et al. 2009) and will not be discussed here. The objective is to evaluate irrigation performance for a combination of inputs, and thus to describe the final infiltration profile and the degree to which the irrigation requirement is satisfied, and to characterize losses by deep percolation and runoff. The model assumes one-dimensional surface flow, which limits the analysis to a single furrow, border strip, or basin. Furthermore, in the case of borders and basins, the model cannot deal with the effect of slope across the width of the field and assumes that inflow is distributed uniformly along the upstream boundary.

[1]Research Hydraulic Engineer, U.S. Dept. of Agriculture, Agricultural Research Service (USDA-ARS), U.S. Arid Land Agricultural Research Center, 21881 N. Cardon Ln., Maricopa, AZ 85138 (corresponding author). E-mail: eduardo.bautista@ars.usda.gov

[2]Information Technology Specialist, U.S. Dept. of Agriculture, Agricultural Research Service (USDA-ARS), U.S. Arid Land Agricultural Research Center, 21881 N. Cardon Ln., Maricopa, AZ 85138. E-mail: james.schlegel@ars.usda.gov

[3]Senior Hydraulic Engineer, West Consultants, Inc., 8950 S 52nd St. # 210, Tempe, AZ 85284; formerly, Director, U.S. Dept. of Agriculture, Agricultural Research Service (USDA-ARS), Arid Land Agricultural Research Center, 21881 N. Cardon Ln., Maricopa, AZ 85138. E-mail: bclemmens@westconsultants.com

Note. This manuscript was submitted on September 30, 2014; approved on June 23, 2015; published online on August 18, 2015. Discussion period open until January 18, 2016; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Irrigation and Drainage Engineering***, © ASCE, ISSN 0733-9437/04015038(11)/$25.00.

SRFR originated from surface irrigation models developed in the late 1970s (Strelkoff and Katopodes 1977a; Katopodes and Strelkoff 1977). These research models were programmed in the *FORTRAN IV* language, using concepts in software architecture that are now more than 40 years old. These research programs provided the foundation for the *BRDRFLW* modeling system (Strelkoff 1985). Like its predecessors, *BRDRFLW* was developed for mainframe computers and programmed in *FORTRAN*. While the program dealt only with border irrigation, it allowed the user to select between the zero-inertia and kinematic-wave solution models, and furthermore provided various options for calculating hydraulic resistance and infiltration and for specifying the bottom profile. With the advent of personal computers, *SRFR* was introduced in 1990 (Strelkoff 1990, 1991) as an end-user application under the *DOS* operating system. The program added various options, including the ability to model furrow and basin irrigation systems, in addition to borders, and a graphical user interface.

SRFR evolved over the next 15 years, partly through addition of configuration options, but also through computational improvements. Early versions of the software were prone to computational failures and could not deal with many complicated flow conditions that can arise in practice. For example, numerical solutions of the governing equations can become unstable and eventually converge to negative flow depths at one or more computational nodes along the field. Inflow rate variations can induce the irrigation stream to contract from the downstream end, stop, and eventually readvance over previously wetted soil. The ability to overcome computational incidents and to adjust computations in response to unexpected changes in boundary conditions substantially improved during this period (Strelkoff 1993).

As new user options and routines for managing the computations were introduced, the original code underwent reprogramming over the years to achieve greater modularity, but fundamental elements of the software architecture were preserved because of programming language limitations. With numerous new features being planned for future versions of the software, including options for modeling constituent transport and new options for modeling infiltration, their development was being hindered by the obsolete architecture of the *SRFR 4* code.

An upgraded engine, *SRFR 5*, was released with *WinSRFR 4.1*. The new engine was developed using an object-oriented architecture in Microsoft's *.NET* development environment. New configuration options and computational improvements have been introduced since its initial release. This paper discusses the architecture of the *SRFR 5* modeling system and its object-oriented implementation.

Governing Equations

Surface flow in surface irrigation can be modeled with modified versions of the de Saint Venant (1871) equations, which describe continuity and momentum for one-dimensional unsteady open-channel flow. In differential form, these equations are (Strelkoff and Clemmens 2007)

$$\frac{\partial A_y}{\partial t} + \frac{\partial Q}{\partial x} + \frac{\partial A_Z}{\partial t} = 0 \tag{1}$$

$$\left[\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{A_y}\right) + \frac{v}{2}\frac{\partial A_Z}{\partial t}\right] + gA_y\left(\frac{\partial y}{\partial x} - S_0 + S_f\right) = 0 \tag{2}$$

Eq. (1) expresses the principles of conservation of mass and states that the time t [T] rate of change in the surface volume per unit length A_y [L^3/L] is a function of the rate of change with distance x [L] of water flux Q [L^3/T] and the time rate of change of water loss due to infiltration A_z [L^3/L]. A_z represents the subsurface flow component and will be briefly discussed in the following paragraph. Conservation of momentum is expressed by Eq. (2), which states that the time rate of change of momentum (represented by the term $\partial Q/\partial t$) is a function of the distance rate of change of momentum flux [$\partial/\partial x(Q^2/A_y)$], the sum of forces acting on the flow, and momentum sinks. The forces are the gradient in hydrostatic pressure ($gA_y\partial y/\partial x$), the weight component of the water in the direction of flow (gA_yS_0), and frictional resistance by the soil surface and vegetation (gA_yS_f), in which g is the acceleration of gravity [L/T^2], y the flow depth, S_0 [-] the field bottom slope, and S_f [-] the friction slope. The sink term $v/2 \cdot \partial A_Z/\partial t$ represents the loss of momentum due to infiltration, with v = flow velocity [L/T] (Strelkoff 1969). Eq. (2) is written in nondivergent form, meaning that under some conditions, it fails to properly conserve momentum.

Historically, *SRFR* has used empirical functions to represent the infiltration process. However, since the *SRFR* modeling system aims to be both a practical and research tool, it is being developed to be compatible with different infiltration modeling options, under the assumption that the choice for a particular modeling task depends on the needs of the user and the availability of data. Hence, the primary focus of the *SRFR* development efforts, and the focus of this article, is the surface flow component.

Eqs. (1) and (2) constitute a coupled system of hyperbolic, nonlinear differential equations. With hyperbolic equations, computations start with known values for the dependent variables at time zero, and the solution then propagates forward in time along paths in the x-t plane known as characteristics. Since the speed of propagation along the characteristics is finite, the solution at any point in the computational domain depends on a finite region bounded by characteristics intersecting at that point. Eqs. (1) and (2) are associated with two sets of characteristics, one with a positive slope (i.e., positive speed), while the other can have a negative, zero, or positive slope. Information entering the computational domain through a positive characteristic can only travel in the positive (downstream) direction, while information entering through a negative characteristic travels upstream. This allows modeling of waves reflecting off the downstream boundary. The overall behavior of hyperbolic equations needs to be taken into account when developing numerical solutions in order to prevent computations from becoming unstable or producing inaccurate results. Numerical schemes based on the method of characteristics, explicit finite differences, implicit finite differences, and finite volumes have been proposed for solving the unsteady open-channel flow equations. The fact that computational methods for the unsteady flow equations continue to be an area of active research attests to the mathematical challenges presented by these equations.

Mainly because of computational difficulties, simplified versions of Eq. (2) were adopted for the *SRFR* development. Katopodes and Strelkoff (1977) showed that under typical irrigation conditions, where flow velocities and the Froude number of the flow are small (typically less than 0.2), the zero-inertia model can produce solutions similar to those computed with the hydrodynamic model. The zero-inertia model neglects the inertial terms in the momentum equation (as well as the infiltration contribution) and, thus, replaces Eq. (2) with

$$gA_y\left(\frac{\partial y}{\partial x} - S_0 + S_f\right) = 0 \tag{3}$$

Eqs. (1) and (3) represent a parabolic system of differential equations. As with hyperbolic equations, information propagates along characteristic paths, but the speed of propagation is infinite. Hence, the domain of dependence at a point in the x-t plane is the entire computational domain bounded by the current time line. Because of this property, parabolic equations can be solved efficiently and with fewer computational problems than hyperbolic ones with implicit finite difference schemes. Those schemes make all computational nodes at the current time line dependent on one another. Since characteristic paths emanate from both boundaries, as in the hydrodynamic model, the zero-inertia equations can model the reflection of waves.

The zero-inertia model sometimes has been labeled as the diffusion-wave model, following the diffusion-wave analogy first proposed by Hayami (1951). However, various researchers (Ponce 1990; Sivalapan et al. 1997) have shown that diffusion wave models containing inertial terms can be developed from modified forms of the hydrodynamic equations. The zero-inertia (also referred to as noninertial) model is, then, a particular type of diffusive wave model.

Numerical solutions of Eqs. (1) and (3) can still run into difficulties. Specifically, they can produce artificial oscillations in the computed depth and discharge profiles when the field bottom slope is relatively large, as the surface profile is relatively uniform but experiences very rapid changes near the tip of the advancing wave. Under those conditions, since the weight of the fluid is essentially in balance with the frictional resistance, the pressure gradient term is small and can be neglected. This reduces Eq. (2) to

$$S_f = S_0 \tag{4}$$

which implies normal flow depth at any distance x as a function of the local Q. The Chezy equation is typically used to compute friction slope in open channel flow

$$S_f = \frac{v^2}{C^2 R} \tag{5}$$

in which C is the Chezy coefficient [$L^{1/2}/T$], R the hydraulic radius [L], and v is as previously defined. In *SRFR*, the default option for calculating C is with the Manning (1889) equation but the Sayre and Albertson (1961) equation is provided as an advanced option. Different authors have discussed the merits and limitations of those

formulations when modeling surface flow in surface irrigation systems (e.g., Maheshwari 1992; Trout 1992).

Eq. (1) in combination with Eqs. (4) and (5) is known as the kinematic-wave model (Lighthill and Whitham 1955) and consists of a single hyperbolic partial differential equation with one dependent variable. The model is associated with positive characteristics only, and thus, with waves that travel in the downstream direction only. Since boundary conditions can only be specified at the upstream end of the field, the kinematic-wave model cannot be used with blocked-end systems. The model cannot be used either when the bottom slope is zero or adverse on any field segment, conditions under which normal flow cannot be attained. Katopodes and Strelkoff (1977) recommend using the kinematic-wave model when the value of a dimensionless parameter P, calculated as a function of slope, normal depth, inflow rate, and the time needed to infiltrate a depth equal to the normal depth, is greater than 100. This condition is more easily met with increasing values of slopes and irrigation time.

SRFR simulates irrigation problems using either the zero-inertia or kinematic-wave model. The parent application, *WinSRFR*, selects a method for the particular data, taking into account the downstream boundary condition and field slope. The USDA Natural Resources Conservation Service border design methodology (USDA-SCS 1974) assumes normal depth for upstream flow depth calculations when the field bottom slope is greater or equal to 0.004. The kinematic-wave model is selected under these slope conditions, while the zero-inertia model is selected under any slope condition, whenever the downstream end is blocked. The user can override the selected model with free-draining systems, subject to the constraint on zero or adverse slopes.

Initial and Boundary Conditions

In principle, A_y and Q are the dependent variables in Eqs. (1) and (3); however, the *SRFR* modeling system actually works with y and Q. Evidently, y and A_y are uniquely related through the user-specified cross-sectional definition. Therefore, conditions at $t = 0$ are given by

$$y(x,0) = 0 \qquad Q(x,0) = 0 \tag{6}$$

Upstream boundary conditions that can be used in combination with the zero-inertia and kinematic-wave models are a specified flow rate or specified flow depth hydrograph as a function of time

$$Q(0,t) = \begin{cases} Q_0(t), & t \le t_{co} \\ 0, & t > t_{co} \end{cases} \tag{7}$$

$$y(0,t) = y_0(t), \quad \forall\ t \tag{8}$$

$$Q(0,t) = Q_0(t), \quad t \le t_{co} \qquad y(0,t) = y_0(t), \quad t > t_{co} \tag{9}$$

In the above expressions, t_{co} is the cutoff time [T]. *SRFR* handles most irrigation events with Eq. (7). Currently, Eq. (8) is not used. Eq. (9) is used to simulate drainback irrigation, a method in which the surface water stored in a basin flows back into the supply channel after cutoff, thus increasing the inflow rate to a basin located immediately downstream. The outflow from the upstream basin is determined by the water level drawdown in the supply channel. The rate at which the water level drops has to be specified by the user.

In the field, the upstream flow depth will decrease gradually until recession time. This process is adequately modeled with the zero-inertia equations, but not with kinematic-wave theory, as the model implies an instantaneous drop in water depth at cutoff time. This assumption is not unreasonable when the field slope is large, for which a short lag time between cutoff and recession time at the boundary can be expected. However, the instantaneous recession can create artificial oscillations in the computed profile. To avoid this problem, *SRFR* approximates the gradual flow depth decrease at the boundary by combining zero-inertia and kinematic-wave calculations (Clemmens and Strelkoff 2011).

Downstream boundary conditions can only be specified with the zero-inertia models. The following condition applies during advance:

$$y(x_a,t) = 0; \qquad Q(x_a,t) = 0, \quad t \le t_L \tag{10}$$

with x_a = advance distance; t_L = final advance time [T]; and L = field length. After the stream reaches the end of the field, water runs off if the field end is open or ponds if blocked. With free-draining systems, a free overfall condition is assumed at the downstream boundary. Strelkoff and Katopodes (1977b) determined that with the zero-inertia model, the theoretically correct boundary condition to apply at a free overfall is $y = 0$, since the specific energy is a function of depth alone and the minimum is attained with zero depth. The resulting value of discharge remains bounded. This boundary condition was first used by Strelkoff and Katopodes and subsequently in *BRDRFLW* and *SRFR*. This approach complicates the computation of infiltration at the boundary with formulations that are flow-depth dependent (e.g., the Green-Ampt formula), while having no effect on the computed runoff, just on the shape of the surface profile near the boundary. Thus a more general expression for the boundary condition is

$$y(L,t) = f(Q), \quad t > t_L \tag{11}$$

where $f(Q)$ = critical flow relationship, as in the hydrodynamic model, or normal depth if the slope is steep (conditions under which use of the kinematic-wave model would be advisable).

If the field end is blocked, then the downstream boundary condition is simply

$$Q(L,t) = 0, \quad t > t_L \tag{12}$$

During an irrigation event, depending on the relationship between inflow and infiltration rate at a particular time, the advancing front may slow down, stop, and retreat. This front-end-recession condition can happen during advance or after advance is complete, prior to cutoff. It can also happen with surge irrigation, depending on the discrete pulses of applied water. In those cases, a second surge may be introduced into the field while the initial surge may still be advancing, and may overtake and merge with the initial surge. The timing and location of these events cannot be determined a priori and have to be resolved as part of the solution.

Solution Method: Numerical Scheme

SRFR uses a four-point scheme based on Eulerian integration to solve both the zero-inertia and kinematic-wave equations. This scheme is explained with the help of Fig. 1. Values for the dependent variables are known at time line t and a solution is sought at timeline t_{i+1}, with the solution advancing by a time increment $\delta t = t_{i+1} - t_i$. The computational domain consists of N rectangular cells, with each cell representing a control volume. Each cell is defined by two nodes at t_i and two nodes at t_{i+1}, with both sets of nodes located at the same distance grid points, x_k and x_{k+1}. For convenience, the letters L and R are used to represent, respectively, the left (x_k) and right (x_{k+1}) nodes at time t_{i+1}, while the subscripts J and M represent the left and right nodes at time t_i (Walker and Skogerboe 1987). Thus, the length of an individual cell is identified in the diagram δx_{LR}. With Eulerian integration, the advancing

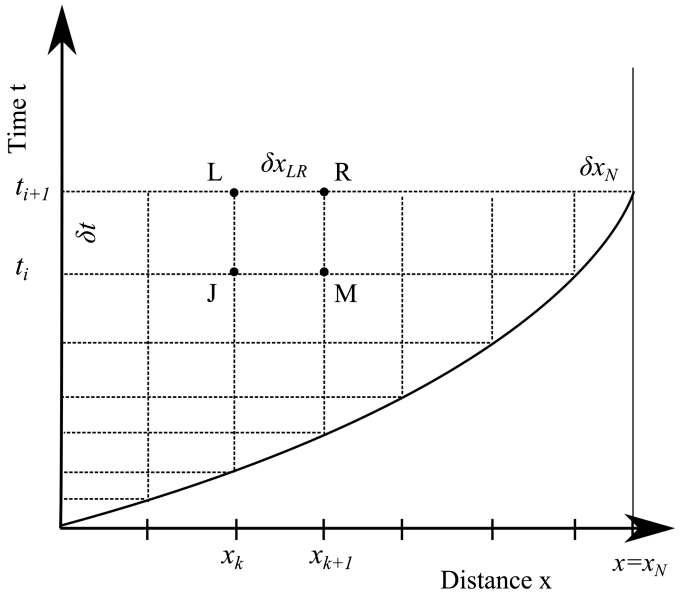

Fig. 1. Computational cell with the four-point scheme and the resulting computational grid

stream is represented by adding a cell to the downstream end of the computational domain, with all cells stationary. In the following discussion, δx_N is the value of δx_{LR} for the tip cell, and the position of the advancing front is x_N. The four-point scheme expresses the conserved quantities as a function of weighted averages of the dependent variables at the four nodes that define the cell. Details on the derivation based on the integral relationships for the governing equations can be found in the previously mentioned references.

Application of the box scheme to the equation of conservation of mass yields

$$[\theta(Q_L - Q_R) + (1-\theta)(Q_J - Q_M)]\delta t - \{[\phi_y(A_{yL} - A_{yJ}) + (1-\phi_y)(A_{yR} - A_{yM})]\}\delta x_{LR} - \{[\phi_z(A_{zL} - A_{zJ}) + (1-\phi_z)(A_{zR} - A_{zM})]\}\delta x_{LR} = R_c \quad (13)$$

The equation is satisfied when R_C, the residual of conservation of mass, is equal to zero. In this expression, θ is a temporal weighting factor and ϕ_y and ϕ_z are spatial weighting factors for the surface and subsurface volumes, respectively, all of which have a value between 0 and 1. These parameters, together with the relationship between temporal and spatial step size and the Courant number of the flow, affect the numerical stability of the box scheme when applied to the hydrodynamic equations (e.g., Cunge et al. 1980; Lyn and Goodwin 1987). In that case, the solution is unconditionally stable when $\phi_y = \phi_z = ½$ and $0.5 < \theta \leq 1.0$, the case in which the finite difference approximation is known as the Preissman scheme. These values have proven adequate when applied to the zero-inertia equations, with θ set to 0.6. Prior versions of *SRFR* adopted a scheme for calculating individual ϕ_z for each interior cell, taking into account the curvature of the infiltrated profile (Strelkoff and Clemmens 2007; Strelkoff et al. 2012). Computational tests have shown that such an approach produces volume balance errors of the same magnitude as those computed with $\phi_z = ½$ while being computationally expensive. Therefore, that method was abandoned.

The discrete expression for Eq. (3) follows the original approach of Strelkoff and Katopodes (1977a), who expressed the residual of conservation of momentum, R_M, as a function of variables at time t_{i+1} only ($\theta = 1$). The equation is written using the water surface elevation h instead of y, to facilitate modeling of horizontal water surfaces in field depressions (Strelkoff and Clemmens 2007; Cunge et al. 1980)

$$[\phi_y A_{yL} + (1-\phi_y)A_{yR}](h_L - h_R - S_{fLR}\cdot\delta x_{LR}) = R_M \quad (14)$$

As with Eq. (13), Eq. (14) is satisfied when $R_M = 0$.

Strelkoff and Katopodes (1977a) derived special spatial weighting factors for the tip cell (shape factors) to account for the increasing curvature of the surface and subsurface profile increases near the tip of the advancing wave. These surface and subsurface shape factors are uncertain because their derivation is subject to numerous assumptions and depend on the roughness and infiltration formulation used in the analysis (e.g., Walker and Skoberboe 1987; Zerihun et al. 2005). However, it is easy to show through computational tests that zero-inertia model solutions are sensitive to these shape factors mostly during the computation of the initial time steps. Later in the calculations, their effect is spread out over all computational nodes at the current time line. Thus, those factors are derived in *SRFR* using the equations originally presented in Katopodes and Strelkoff (1977).

Application of Eqs. (13) and (14) to N computational cells produces a system of $2N + 2$ nonlinear algebraic equations with at least $2N + 4$ unknowns—the values of the dependent variables at time $i + 1$ at each node. Equations and unknowns are balanced by applying two boundary conditions, typically an inflow rate at the upstream (left) boundary and one of several conditions at the downstream (right) boundary. During advance, the depth and discharge are zero at the right boundary, but the advance distance at a given time or the advance time at a given distance become the unknown. During runoff with the zero-inertia conditions, the depth is zero and the discharge is unknown. An alternative is to specify a depth-discharge relationship, where one of these variables is considered unknown and the other is expressed as a function of the unknown variable. With a blocked end, the discharge is zero and the depth is unknown.

The system of equations is solved iteratively using the Newton-Raphson method. To this end, Eqs. (13) and (14) are linearized using a Taylor series expansion

$$\frac{\partial R_C}{\partial A_L}\Delta y_L + \frac{\partial R_C}{\partial Q_L}\Delta Q_L + \frac{\partial R_C}{\partial A_R}\Delta y_R + \frac{\partial R_C}{\partial Q_R}\Delta Q_R + \frac{\partial R_C}{\partial \Psi}\Delta\Psi = -R_C \quad (15)$$

$$\frac{\partial R_M}{\partial A_L}\Delta y_L + \frac{\partial R_M}{\partial Q_L}\Delta Q_L + \frac{\partial R_M}{\partial A_R}\Delta y_R + \frac{\partial R_M}{\partial Q_R}\Delta Q_L + \frac{\partial R_M}{\partial \Psi}\Delta\Psi_L = -R_M \quad (16)$$

In the above expressions, the derivative terms are elements of the Jacobian matrix for the vector $[R_C \quad R_M]^T$, the Δ terms are the unknown incremental corrections to the values of the dependent variables, and Ψ represents either an unknown δt or δx_N and applies only for advance calculations. The resulting system of linear equations can be represented as

$$\nabla \boldsymbol{R}\cdot\boldsymbol{\Delta} = -\boldsymbol{R} \quad (17)$$

in which R = vector of residuals for all cells; Δ = vector of corrections; and ∇R = Jacobian matrix for R. The system is sparse and banded if the downstream boundary condition is $y = f(Q)$ or if δx_N needs to be computed for a specified δt. With this structure, the matrix can be solved with the conventional double-sweep

technique (Cunge et al. 1980; Walker and Skobergoe 1987; Strelkoff 1992). The double-sweep method uses recursive relations developed by assuming a linear relationship between a Δy and its corresponding ΔQ

$$\Delta Q_{k-1} = E_k \Delta y_{k-1} + F_k \tag{18}$$

The recursion formulas are built node by node by combining Eq. (18) with Eq. (15) to obtain a relationship for Δy_{k-1} and then combining the result with Eqs. (16) and (18) again, but written for node k to obtain a relationship for Δy_k. The procedure is initialized by assuming $E_1 = 0$ and $F_1 = \Delta Q_0$. The downstream boundary condition is used to complete the downstream sweep, from which the correction for the last dependent variable is determined. A second set of recursive relationships then calculates the corrections going in the upstream direction. The recursive relations are detailed in the aforementioned references.

If δx_N is specified and δt the unknown, which is the default method for computing advance, then the matrix is still banded but the last column contains the terms representing the derivatives of R_C and R_M with respect to time. For those cases, the system of equations is solved using a modified double-sweep method (Strelkoff 1992), developed with Eq. (19) in lieu of Eq. (18)

$$\Delta Q_{k-1} = E_k \Delta y_{k-1} + H_k \Psi + F_k \tag{19}$$

The resulting corrections Δ are used to update the dependent variables and then the values of the residuals. The solution converges when the absolute value of the residuals falls below a tolerance value. If not, the process is repeated.

Eq. (13), in combination with a normal depth relationship (e.g., the Manning equation) is used to solve the kinematic-wave equation. However, in contrast with the zero-inertia application, ϕ_y is set to zero to reduce artificial oscillations that often develop near the downstream boundary. Such an approach has been suggested by Lyn and Goodwin (1987), when the box scheme is applied to the hydrodynamic equations, but only when negative waves are expected to have little influence on the solution. Clearly, this recommendation would apply to the kinematic-wave model. During advance, if $\Psi = \delta t$, then the solution is implicit and iterative and it is found, again, using a modified double-sweep method developed from Eq. (15) alone (Strelkoff 1992). Otherwise, the system of N equations is solved cell by cell, starting from the upstream cell where the boundary condition is given.

Whether using the zero-inertia or kinematic-wave equations, the core computational tasks that need to be carried at each time step can be summarized as follows:

- Retrieve pertinent data from the previous time line for all nodes needed for computations at the current time (y, Q, A_y, A_z, h, S_f, etc.);
- Generate initial estimates of y and Q for all nodes at the current time line;
- Compute estimates of nodal variables for the current time line (A_y, A_z, h, S_f, etc.);
- Apply the discrete equations of conservation of mass and momentum to each cell at the current time line;
- Solve the system of equations to obtain new estimates for y and Q at the current time line; and
- Repeat this process until the continuity and momentum equations are satisfied.

Management of the Numerical Solution

Previous research and experience suggest that the numerical algorithms employed by *SRFR* are numerically stable and approximate the solution of the differential equations accurately. Still, the computations can fail for reasons to be explained below. Moreover, successful simulations occasionally produce large volume balance errors and, thus, invalid results. This latter problem received particular attention during the development and testing of *SRFR 5*.

Preventing computational incidents and reducing volume errors depends on how well several contingencies are managed. A critical component of the *SRFR* modeling system is the code that manages those unforeseen circumstances, which can be classified as

- Problems with the solution of the system of equations;
- Discretization problems (i.e., problems with the spatial or temporal increment used to advance the solution in time); and
- Boundary condition problems.

Time step calculations begin by examining the state of the computational domain. The code identifies appropriate boundary conditions and a discretization strategy from the problem data and the history of calculations. *SRFR* then attempts to solve the time step. The calculations are hierarchical and computational incidents can occur at different levels of this hierarchy. The bottom level in this hierarchy is, of course, the linearized equation system solver based on double–sweep algorithm. In the following discussion, this equation solver will be identified as EQSWP. The code attempts to identify conditions that would lead to computational incidents and generates an error condition—an exception in computer programming jargon—that pushes the handling of the problem to a higher level in the calculation hierarchy. This results in modifications to the solutions generated by EQSWP, the discretization, and/or the boundary conditions. Error objects, which are vital components of modern object-oriented programming languages, are extensively used by *SRFR 5* to handle exceptions.

Problems with the Solution of the System of Equations

Problems at this level are (1) division by zero, (2) oscillations in the calculation of the residuals, (3) an iteration that produces unacceptable values for the unknown variables, and (4) failure to converge.

A division-by-zero problem occurs at the EQSWP level. It indicates that the discrete governing equations cannot be solved with the current data and that a change in spatial or temporal increment is needed. For example, the problem can arise with a pronounced deceleration of the flow, and difficulties in getting the water to advance to the next prescribed advance increment. New calculations have to be attempted, but using a smaller space increment.

Calculations often deal with very small changes in the values of the dependent variables. Because of numerical processor limitations, the computed residuals can oscillate. Typically, this problem involves only the momentum residual of the zero-inertia model and is confined to one or a few cells, and the residual is within an order of magnitude of the tolerance value. This problem is addressed by monitoring the history of computed residuals and accepting the solution if oscillations are detected and the convergence criterion is nearly met.

At some point in the simulation, the calculations will produce negative flow depths. Most often, those negative values signal the dewatering of a node and, thus, the need to change the location of a boundary and, perhaps, the corresponding boundary condition. However, negative flow depths can also be computed during an EQSWP iteration and may represent an overcorrection. For example, when simulating advance, artificial oscillations can arise in the flow depth profile during the course of an apparently smooth simulation, most frequently near the boundaries. While these oscillations attenuate with time, their initial calculation may be problematic. Abrupt changes in boundary conditions and field properties can also induce large initial corrections. *SRFR* attempts to address

this type of problem, only while the stream is advancing, by scanning the sign of the resulting y_{i+1} values. If at least one negative value is found, then all Δ's are progressively reduced. If $\Delta\delta t$ is negative, then the adjustments continue until δt becomes unrealistically small. At that point, new time step calculations are attempted using a smaller δx_N.

The Newton-Raphson scheme generally converges to the solution or produces unacceptable results in just a few iterations. Occasionally, the scheme will converge slowly without producing substantial improvement in the value of the residuals, as a result of discretization or boundary conditions problems. Hence, an exception needs to be generated when the number of iterations is excessive.

Discretization Problems

The computational grid needs to be managed to assure accurate numerical integration and to prevent or resolve computational incidents. For accurate integration, the subintegrand functions must vary smoothly over the length of a cell, without discontinuities in the value of the dependent variables. To this end, setup code determines the location of computational nodes in the x-t plane as a function of the problem data and establishes discretization guidelines. The solution progresses by using either a δx or δt increment. The type of increment to use and the magnitude of that increment are determined from calculations at each time step, in response to results from previous time steps or to problems with the current step.

The spatial discretization is initially defined by the location of so-called *hard* x-nodes. These are locations where field properties (channel bottom elevations, cross sections, infiltration, and roughness) are specified, or where a change in boundary condition is specified (e.g., cutoff by distance). Since these hard nodes define boundaries of computational cells, they must not be specified in very close proximity to one another because that will result in a very small δx_N and potential convergence problems. Likewise, hard t-nodes are needed to precisely enforce changing boundary conditions. The setup code determines an initial spatial discretization based on the list of spatial hard nodes and the default advance increment δx_D, where δx_D = field length/cell density. The cell density is set by the *WinSRFR* user interface based on the problem data, generally to a value between 40 and 80 cells independently of the field length, but that value can be overridden by users. Each simulation model imposes additional spatial discretization requirements. The kinematic-wave model requires a finer cell resolution at the head of the field to handle postcutoff recession calculations with the zero-inertia/kinematic-wave approximation described earlier, while the zero-inertia model requires a denser grid near the downstream boundary to handle potentially large δt changes at the beginning of postadvance calculations.

Using the setup information, calculations for a time step begin by setting targets for the advance distance (x_T) and time t_{i+1} (t_T), independently of whether the time step is eventually going to be calculated with given values of δx_N or δt. Those targets are determined based on the list of hard nodes, δx_D, and the history of past spatial and temporal increments. For example, x_T could be the location of a hard x-node, if that node is located between x_{N-1} and $x_{N-1} + \delta x_D$. Various empirical rules are employed to try to keep consecutive cells from being too dissimilar in length. If a hard t-node exists and that time has not been reached, then that value would determine t_T. When calculating advance with δx_N given, the solution tries to advance to x_T. If the resulting t_{i+1} is greater than t_T, calculations are started over and rerun with δt given ($\delta t = t_T - t_i$) and δx_N unknown. Likewise, if the advance is being calculated with δt given, and the stream has not reached the end of the field, then calculations for the time step are repeated if the resulting x_N exceeds x_T.

As explained in the previous section, computational incidents often are resolved by repeating the calculations for a time step with a smaller spatial or temporal increment. This is not however the primary mechanism controlling the step size. When the stream is advancing, the code monitors the deceleration of the stream using the ratios $\delta t/\delta x_N$ and t_{i+1}/x_N. If the flow is slowing down, empirical rules are used to preemptively reduce δx_N for the next time step. This strategy is particularly useful for resolving problems that can build up over several time steps and that will eventually lead to termination of the execution. This mechanism also allows the target δx_N to increase for subsequent time steps, up to the value defined by the default cell width, δx_D, if the flow accelerates. Likewise, if δt is specified for the current calculations, the magnitude of δt is determined using empirical rules that account for the past history of time increments. These rules allow the step to increase or decrease, as needed. It is important to note that the mechanisms described above control the discretization relative to δx_D, which is a function of the *WinSRFR*-selected (or user-selected) cell density. While experience indicates that the rules used by *WinSRFR* to select a cell density work well most of the time, the step size still can be too coarse for some problems, which makes the solution sensitive to the step size, albeit slightly. Thus, it is always good practice for a modeling task to test the sensitivity of a selected solution to cell density (i.e., coarse, fine, extra fine).

Boundary Condition Problems

As noted earlier, time step calculations can begin with uncertain knowledge of boundary locations and conditions at time t_{i+1}. Those data have to be assumed based on conditions at t_i, and revised with calculation outcomes. For example, water may cover the entire field after cutoff. That condition might persist for one or more time steps and then recession will begin. Recession calculations will then eliminate one or more computational nodes during a time step, perhaps beginning at the upstream end of the field. Determining when recession begins and how many nodes to eliminate during a time step involves trial and error, until the equations can be satisfied for a defined stream.

Determining appropriate boundary locations and conditions may be relatively straightforward when the field data and boundary conditions are uniform but can get very complicated when the flow exhibits significant deceleration and/or when the problem data are variable. For example, based on previous results and problem data, it may be reasonable to assume that the stream is advancing during the current time step. Calculations begin, perhaps, with $\delta x_N = \delta x_D$. However, as noted in previous sections, calculations need to account for the t_T targets, to properly enforce changes in boundary condition data. Thus, calculations may need to switch from advance by distance to advance by time. On the other hand, calculations may run into trouble when advancing by distance, perhaps because the distance increment is too large, or because the front of the wave is not advancing or even receding due to inflow reductions.

The hierarchy of calculations that deals with testing different boundary locations and conditions is built around the computational options (solution modes) of EQSWP. These modes define the setup of Eq. (17)—boundary conditions and unknowns. With the zero-inertia model, those solution modes are

- Advance-by-time (δt known, δx_N solved for);
- Advance-by-distance (δx_N known, δt solved for);
- Runoff-by-time (δt known, δx_N not applicable); and
- Pond-by-time (δt known, δx_N not applicable).

Upstream	Downstream
Inflow	Advance
No Inflow - Depletion	Runoff
No Inflow - Recession	Ponding
Drainback	Recession
	Re-Advance

Fig. 2. Upstream/downstream boundary condition combinations used by SRFR to solve a stream

Because of underlying solution assumptions, the ponding-by-time mode does not apply to the kinematic-wave model. These EQSWP modes are applied to the various possible boundary condition combinations for a stream. *SRFR* handles 4 types of upstream boundary conditions and 5 downstream boundary conditions (Fig. 2), for a total of 20 upstream/downstream boundary combinations. Note that code allows the upstream drainback condition to be used in combination with all downstream boundary conditions even though the ponding and recession conditions are the only ones that are likely to be used in practice. Boundary condition combinations that involve an advancing stream or readvancing stream can be solved with either the advance-by-time or advance-by-distance EQSWP modes while other combinations can only be solved with the runoff-by-time or pond-by-time modes.

A higher level of the time step calculations involves testing and handling multiple streams. This is clearly the case with surge irrigation, for which computations need to allow for the merging of streams during a time step. Multiple streams can also occur as a result of variable, but nonsurge, inflow to a field with undulating bottom elevations. In those cases, calculations need to allow the stream to split, with one substream receding, while the other one possibly continues to advance.

As with problems with the equation solver or the discretization, the code tries to handle boundary problems preemptively. Time step calculations begin by locating and *conditioning* the stream. This latter process involves scanning the flow depths calculated during the previous time step, and infiltrating water in place for nodes with extremely small flow depths, if justified by the prevailing boundary conditions. Conditioning streams also may involve merging streams, as could occur with surge irrigation.

Implementation in an Object-Oriented Architecture

This section aims to provide an overview of the object-oriented architecture of *SRFR 5*. Additional details are provided in documents prepared in support of the development of the code (*SRFR 5.2 architecture and high-level design*).

The following software development objectives were defined in reprogramming *SRFR*:

- Develop the code in an object-oriented architecture, making full use of the concepts of encapsulation, inheritance, and polymorphism;
- Develop code that is modular, extensible, and maintainable such that new computational capabilities can be added without affecting existing functionalities;
- Develop *SRFR* as a reusable dynamic link library (DLL) that can be made available to other applications and expose its internal mathematical algorithms and functions to client applications through an application programming interface (API); and
- Develop programmer-friendly debugging and diagnostic tools.

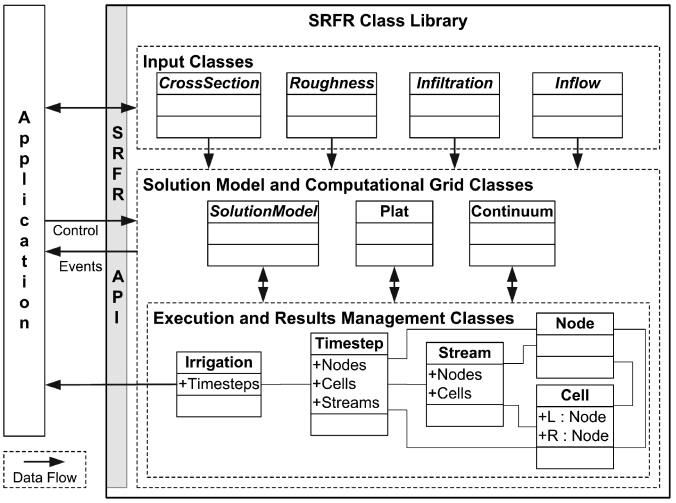

Fig. 3. The SRFR 5 class library

Fig. 3 depicts the class library of *SRFR 5*. In the figure, the classes inside dotted boxes represent groups of related classes—those that manage input data, those that manage computational model and grid data, and those that manage the flow of execution and the resulting outputs. The *SRFR* API is the object that communicates with the client application (e.g., *WinSRFR*). The client application defines through the API the problem data and solution model to use. Outputs generated by Execution and Results Management classes are then made available to the application again through the API.

In accordance with principles of object-oriented programming, computational options are handled, whenever possible, through the use of base classes and subclasses. The base class defines the fundamental data requirements, methods, and outputs for a class, while the subclass defines specific implementations, i.e., the specific option. The subclass provides structures for data storage and methods that are specific to the option. When *SRFR* makes a call to a class, the base class is always used. The client application instantiates the specific (subclassed) objects needed for a simulation and passes references to those objects to *SRFR*. This facilitates writing code that handles multiple options, as well as options that have yet to be developed.

Examples of the above described paradigm are the zero-inertia and kinematic-wave models, which are implemented as subclasses of the SolutionModel base class (identified in Fig. 3 in italics). This base class produces the corrections needed by the Newton-Raphson scheme, but calculations are different for zero-inertia and kinematic-wave models. All SRFR calls to the simulation engine are made through a generic SolutionModel object. Depending on the data, *WinSRFR* determines which specific model to use. A full hydrodynamic model can be added with this architecture at a future time, in principle without changing the existing code. In addition to the solution model, all input classes were implemented using the class/subclass paradigm (also shown in italics in the diagram).

Classes in the Execution and Results Management grouping are associated with the computational grid structure of Fig. 1 and the flow of data during a simulation. The Irrigation class manages the simulation, from time zero until final recession time. The output is the complete description of the irrigation flow. This class interacts with the Plat and Continuum classes, which are responsible for managing temporal and spatial problem data, and thus the evolving discretization. As the name implies, calculations for individual time

steps are managed by the Timestep class. A time step simulation involves solving one or more streams, and thus the Stream class handles calculations at the stream level. Calls to the SolutionModel class are made at the Stream class level. A stream is defined by multiple cells and nodes; thus cell and node data are handled by the Cell and Node classes, respectively.

Fig. 4 provides an overview of the flow of calculations from the SRFR API to the SolutionModel classes. The class name is noted in bold, to the left of each major block in the figure. The subblocks on the right-hand side of each major block identifies the hierarchy of methods called until reaching the final call to the EQSWP linear equation solver. If the method is unsuccessful, then it will generate an exception. Hence, Fig. 4 also identifies the various computational levels at which exceptions are handled. A method may also return a status report to the next level of calculations, with information that may help identify a next course of action.

The Simulate method in the *SRFR* API class starts the simulation, by verifying data, instantiating the objects needed to manage the computational grid and the Irrigation object, and then by calling the Simulate method of the Irrigation class. Exceptions reported at this level terminate the simulation because all attempts to solve the equations for the given time step have failed. While such exceptions are rare nowadays, they can occur if the data are extremely atypical, for example, with unrealistically large variations in field elevation. The Irrigation.Simulate method instantiates the Timestep objects that need to be used to build the solution, manages the solution in time and space, and calls the SolveTimestep method of the Timestep class. An exception reported at the Timestep class level generally cannot be handled, except when calculations involve the zero-inertia equations and a near-stagnant stream with very small flow depths. Under those conditions, water is simply infiltrated in place, thus successfully concluding the simulation.

Calculations of the Timestep class involve four different layers. The first layer corresponds to the SolveTimestep method, which identifies and conditions the streams within the time step and attempts to solve each stream. If multiple streams are determined, then one is identified as the main stream, while the others are labeled secondary streams. If water is flowing into the field, then the main stream is the stream originating at the inlet. If inflow is zero, the main stream is either a stream that is still flowing or the stream with the least volume. Since the primary stream dictates the δt that will be used to solve all other streams, the SolveTimestep method calls the SolvePrimaryStream method and, if needed the SolveSecondaryStream method, based on the δt resulting from

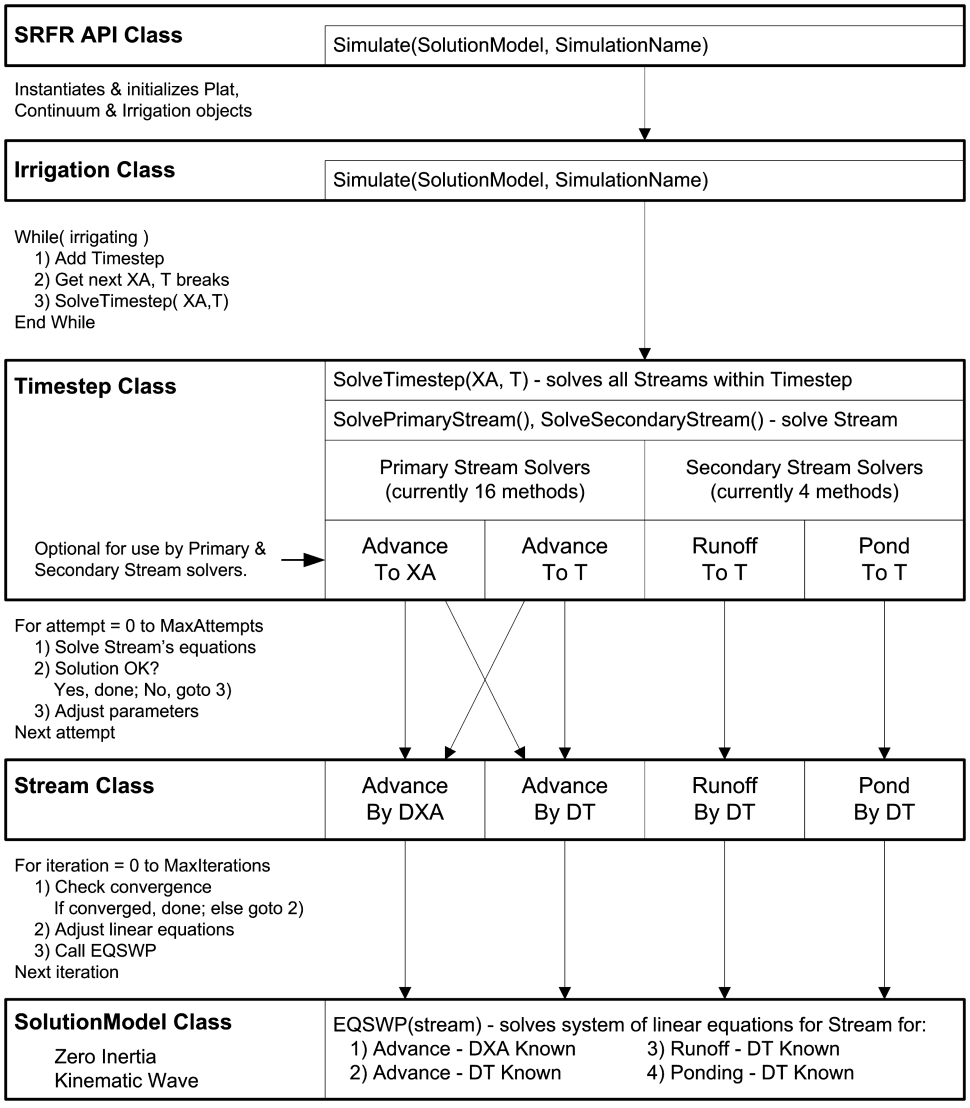

Fig. 4. Simulation flowchart, showing the path from the SRFR API to the EQSWP equation solver

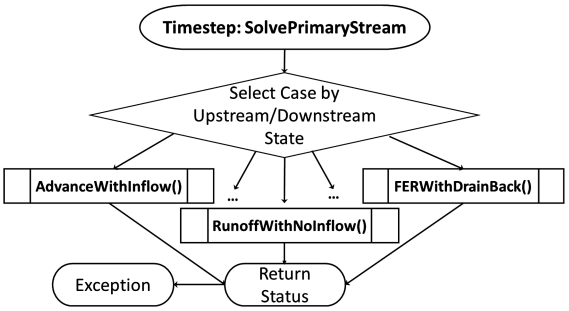

Fig. 5. Flowchart for the SolvePrimaryStream method. The method selects one of the 16 upstream/downstream boundary conditions that can be applied to the primary stream

the primary stream calculations. Fig. 5 illustrates the structure for the SolvePrimaryStream method, and identifies three of the primary stream boundary condition combinations. The name of the first two methods is self-explanatory while the last, FERwithDrainback, applies to downstream-end recession conditions with drainback irrigation. Each of the methods in the second layer selects a stream solver (a method in the third layer), with the selection depending on the boundary conditions and the stream type (primary or secondary). There are 20 different stream solvers, 15 of which apply to the primary stream while 5 apply to secondary streams. Most stream solvers are associated with a specific combination of boundary conditions but some solvers apply to more than one combination. An exception by a primary stream solver signals the need to test an alternative boundary condition combination. Exceptions generated while solving a secondary stream cannot be passed up through the exception hierarchy because δt cannot be modified. Those exceptions are handled within the SolveSecondaryStream method, by infiltrating water in place.

The last layer in the Timestep class, consisting of four methods, is the layer where a stream solution is attempted by using either a time or distance increment. As their names imply, two of those methods are for a stream with a moving downstream boundary, while the other two deal with a stream with fixed or receding boundaries. Unsuccessful calculations generate an exception and trigger a new attempt with a smaller spatial or temporal increment. This process continues until the calculations succeed or the increments become too small, thus generating an exception. The latter event forces calculations back to the level where new boundary conditions are implemented. After calculations succeed, the resulting values of x_A and t_{i+1} are checked against their targets. In the event that the target is exceeded, calculations are repeated but with a different increment, or switched from a specified δx_N to a specified δt or vice versa, as indicated by the vertical and diagonal arrows coming out of the AdvanceToXA and AdvanceToT methods in Fig. 4.

The four methods in the Stream class represent the layer at which the Newton-Raphson scheme is managed. At each iteration, each of these methods calls EQSWP, which is a method of the SolutionModel class (either zero inertia or kinematic wave). The four methods in the Stream class map into one the four EQSWP computational modes. Exceptions generated by EQSWP cannot be handled by the Stream class methods. However, those methods validate the output from EQSWP, and either attempt to adjust the corrections if negative flow depths are computed, or flag nodes that might be undergoing recession. In the latter case, the Newton-Raphson iteration is terminated and the simulation returns to the last computational level in the Timestep class, so that changes can be made for the solution to move forward.

Diagnostics and Code Testing

SRFR 5 replaced all of the text-based diagnostic files of *SRFR 4* with a graphical diagnostic and debugging tool. This tool is available not only to *SRFR* programmers, but also to third-party developers that use the *SRFR* engine. The Simulation Debug Window (Fig. 6) consists of three different viewers. The left-most component, the Irrigation Viewer, is used to inspect the discretization produced by a simulation and, thus, provide a view similar to Fig. 1. Regions in the computational domain with low and high cell density can be easily discerned with this tool. The viewer also identifies cells where the final residuals exceeded the tolerance value. This can be of importance when diagnosing irrigations with volume balance errors. The Stream Viewer, located at the bottom right, is used to inspect graphically and numerically outputs from a stream calculation. By stepping through the streams computed at each time step, one can identify the point at which jagged profiles begin to develop, which can ultimately lead to negative flow depths. The programmer can zoom in on calculations for a specific cell and its associated nodes by clicking on any of the cells depicted in the stream graph. The cell viewer (top right) displays all pertinent information used in the calculation of residuals for the selected cell, including the coefficients on depth and discharge used in the double-sweep solver. The stream, cell, and node results are available for the entire simulation.

The *SRFR 5* code was initially tested by conducting simulations with a suite of benchmarking scenarios. The objective of those tests was to identify and explain differences in results computed with the new code in comparison with the old code. As expected, this process uncovered programming errors in both the new and old code. In addition to this benchmarking process, a code-testing application was developed that runs *SRFR* subject to random inputs. The range of input values to explore is defined prior to each test. Several hundred thousand random runs were executed with this tool with the goal of identifying pathways that would lead to failed simulations. This testing led to substantial enhancements to the structure for handling exceptions. It was not uncommon for up to 10% of the simulations to fail during the initial testing period, and those failures are very uncommon at this stage of the development. The testing program was also used to identify conditions under which calculations produced unacceptable mass balance errors. Likewise, the main sources of those errors have been identified and corrected. Initial testing identified simulations with volume balance errors greater than 20%. Larger errors mostly occurred with simulations involving front-end recession and readvance, and simulations involving larger variations in bottom slope. Currently, the error for a typical simulation is less than 0.1%. *SRFR 5* has been subjected to a more rigorous and extensive testing than any of its predecessors.

Discussion

Key benefits of the current architecture of *SRFR 5* relative to the procedural code of previous versions are worth highlighting. Procedural *FORTRAN* forces all calculations to follow well-defined unambiguous paths. Options are handled with conditional statements, which allow the code to branch to different sets of calculations. The code of *SRFR 4* contains many nested conditional statements to deal with its many options, and thus, it had become extremely difficult to upgrade. As explained earlier, *SRFR 5* takes full advantage of the class/subclass paradigm, and as such, replaces many conditional statements with calls to the base class. This has greatly simplified parts of the code and/or produced greater modularization. As an example, the subroutine that calculated

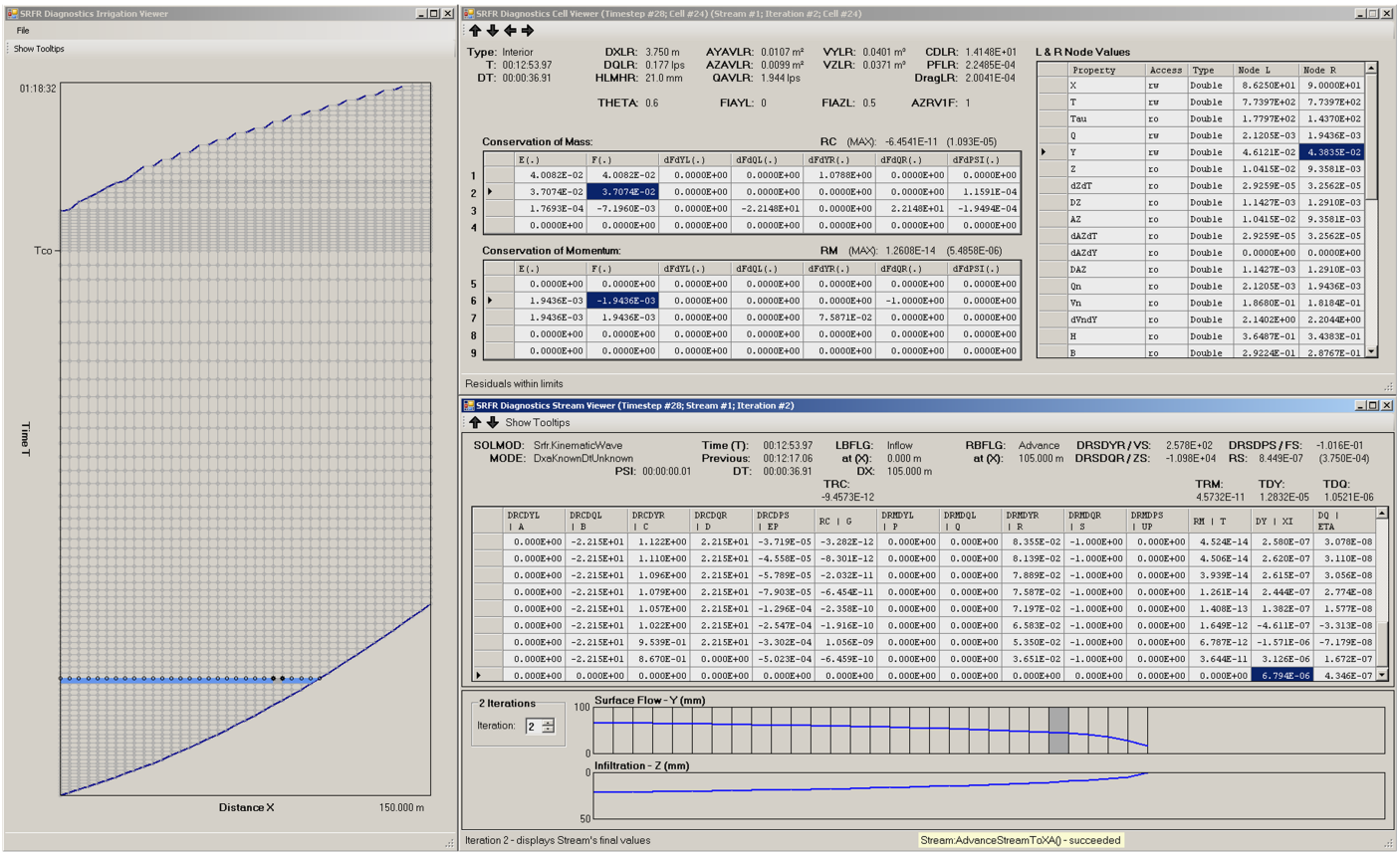

Fig. 6. The simulation debug window showing the irrigation, stream, and cell viewer

the cross-sectional, infiltration, and roughness properties at each node based on y and Q previously consisted of about 800 lines. The current method handles more options than before with slightly less than 100 lines of code.

Future development will test the extensibility of *SRFR 5* classes. Presently, only the Infiltration base class has been tested with recent additions of infiltration options. All infiltration options are implementations of the Infiltration class. Their output is the infiltrated depth and rate for a given opportunity time. *SRFR 5* inherited only empirical infiltration options from *SRFR 4*, and in all of those cases, infiltration is a function of opportunity time only, in addition to the parameters of the formulation. Infiltration based on the Richards and Green-Ampt equations has been added to *SRFR 5*. In addition to opportunity time, these options require a history of flow depths at each computational node. Those implementations required some changes to the architecture of the Infiltration class. Hopefully, future subclasses should be easier to implement and require no further changes to the base class.

An important associated benefit of the class/subclass structure is that subclasses can be developed without having access to the base class code. The authors hope that this feature will encourage third-party development of future *SRFR 5* capabilities. Documentation of the object-oriented architecture is available upon request.

While *SRFR 4* was already relatively robust, *SRFR 5* has stronger capabilities for overcoming computational incidents, partly as a result of computational improvements, but mostly as a result of a modern structure for handling exceptions. Notable computational improvements include the detection of oscillations and the ability to use the governing equations to solve multiple streams. Previously, the solution was applied only to the primary stream, while other streams were solved considering only volume balance or infiltrated in place.

The *SRFR 4* code made extensive use of common blocks for data management. Since that technology provides virtually no mechanisms for data validation, it was easy to write code that used memory segments with no data or inappropriate data (Strelkoff et al. 2000). This problem has been eliminated by encapsulating the data with objects.

As mentioned earlier, *SRFR 5* offers an API that allows a client application to use any of the classes exposed by *SRFR*. In addition, the API provides access to various functions used by *SRFR*. Those functions are independent of the *SRFR* classes and may be useful to a client application. For example, infiltration calculations with a Kostiakov formula are handled by the Kostiakov class, a subclass of the Infiltration class. The actual infiltration calculation is performed by a function in the *SRFR* API. This function is available to *SRFR*, as well as to the client application, but only if the client is developed in the *.NET* environment. An extension to the API has also been developed that allows *SRFR* to communicate with applications developed outside the *.NET* environment, specifically using the Microsoft Component Object Model technology. In this case, the communication is limited to the *SRFR* classes.

Conclusions

The *SRFR* modeling system for surface irrigation was reprogrammed using an object-oriented architecture. The upgraded software is more robust than previous versions and provides a better pathway for adding new modeling capabilities and options. *SRFR*

5 was programmed as a reusable dynamic-link library, exposes its functionality through an application programming interface, and is better suited to support the continued development of the *WinSRFR* software package. The *WinSRFR* software package is available for public download at http://www.ars.usda.gov/services/software/software.htm.

Acknowledgments

The authors would like to acknowledge Dr. Theodor ("Fedja") Strelkoff, who retired as this manuscript was being developed. Dr. Strelkoff was a pioneer in the field of unsteady flow modeling of surface irrigation systems and led the development of the *SRFR* code and its predecessors over three decades. He also contributed some of the initial ideas for this article.

Notation

The following symbols are used in this paper:

A_y = surface volume per unit length [L^3/L];
A_z = infiltration volume per unit length [L^3/L];
C = Chezy coefficient [$L^{1/2}/T$];
g = acceleration of gravity [L/T^2];
L = field length [L];
N = number of computational cells at a given time step;
Q = flow rate [L^3/T];
R = hydraulic radius [L];
R_C, R_M = residuals of conservation of mass and momentum, respectively;
S_0 = field bottom slope [L/L];
S_f = friction slope [L/L];
t = time [T];
t_{co} = cutoff time [T];
t_i = discrete time [T];
t_L = final advance time [T];
v = cross section–averaged flow velocity [L/T];
x = distance [L];
x_a = advance distance [L];
x_k = discrete distance [L];
y = flow depth [L];
Δ = vector of corrections computed by an iteration of the Newton-Raphson method;
∇R = Jacobian matrix for the vector of residuals R;
δt = discrete time increment [T];
δx_{LR} = length of computational cell [L];
δx_N = length of tip cell [L];
θ = temporal weighting factor used by the four-point scheme [-];
ϕ_y, ϕ_z = spatial weighting factors used by the four-point scheme for the surface and subsurface volumes [-]; and
Ψ = unknown variable in the system of equations, either δt or δx_N.

References

Bautista, E., Clemmens, A. J., Strelkoff, T. S., and Schlegel, J. (2009). "Modern analysis of surface irrigation systems with WinSRFR." *Agric. Water Manage.*, 96(7), 1146–1154.

Clemmens, A. J., and Strelkoff, T. S. (2011). "Zero-inertial recession for kinematic-wave model." *J. Irrig. Drain. Eng.*, 10.1061/(ASCE)IR.1943-4774.0000289, 263–266.

Cunge, J. A., Holly, F. M., Jr., and Verwey, A. (1980). *Practical aspects of computation river hydraulics*, The Pitman Press, Bath, England.

de Saint Venant, A. J. C. B. (1871). "Théorie du mouvement non-permanente des eaux avec application aux crues des riviéres et à l'introduction des marées dans leur lit." *Compte Rendus, Acad. Sci.*, 73(148-154), 237–240.

Hayami, S. (1951). "On the propagation of flood waves." *Bulletin No. 1*, Disaster Prevention Research Institute, Kyoto Univ., Kyoto, Japan.

Katopodes, N. D., and Strelkoff, T. (1977). "Dimensionless solutions of border irrigation advance." *J. Irrig. Drain. Div.*, 103(IR4), 401–417.

Lighthill, M. J., and Whitham, G. B. (1955). "On kinematic waves, I. Flood movement in long rivers." *Proc. R. Soc. London, Series A*, 229, 281–316.

Lyn, D. A., and Goodwin, P. (1987). "Stability of a general Preissmann scheme." *J. Hydr. Eng.*, 10.1061/(ASCE)0733-9429(1987)113:1(16), 16–28.

Maheshwari, B. L. (1992). "Suitability of different flow equations and hydraulic resistance parameters for flows in surface irrigation: A review." *Water Resour. Res.*, 28(8), 2059–2066.

Manning, R. (1889). "On the flow of water in open channels and pipes." *Trans. Inst. Civ. Eng.*, 20, 161–207.

Ponce, V. M. (1990). "Generalized diffusion wave equation with inertial effects." *Water Res. Res.*, 26(5), 1099–1101.

Sayre, W. W., and Albertson, M. L. (1961). "Roughness spacing in rigid open channels." *J. Hydr. Div.*, 87(3), 121–149.

Sivalapan, M., Bates, B. C., and Larsen, J. E. (1997). "A generalized, non-linear, diffusion wave equation: Theoretical development and application." *J. Hydrol.*, 192(1–4), 1–16.

Strelkoff, T. (1969). "One-dimensional equations of open channel flow." *J. Hydr. Div.*, 95(HY3), 861–876.

Strelkoff, T. (1985). "BRDRFLW. A mathematical model of border irrigation." Water Conservation Laboratory, Phoenix.

Strelkoff, T. (1990). "SRFR. A computer program for simulating flow in surface irrigation." *WCL Rep. No. 17*, U.S. Water Conservation Laboratory, ARS-USDA, Phoenix.

Strelkoff, T. (1991). "SRFR:A model of surface irrigation—Version 20." *Irrigation and drainage*, W. F. Ritter, ed., ASCE, Honolulu, 676–682.

Strelkoff, T. (1992). "EQSWP: Extended unsteady flow double-sweep equation solver." *J. Hydr. Eng.*, 10.1061/(ASCE)0733-9429(1992)118:5(735), 735–742.

Strelkoff, T., Clemmens, A. J., and Schmidt, B. V. (2000). "ARS software for simulation and design of surface irrigation." *Proc., 4th Decennial National Irrigation Symp.*, ASAE, Phoenix, 290–297.

Strelkoff, T., and Katopodes, N. D. (1977a). "Border-irrigation hydraulics with zero inertia." *J. Irrig. Drain. Div.*, 103(IR3), 325–343.

Strelkoff, T., and Katopodes, N. D. (1977b). "End depth under zero-inertia conditions." *J. Hydr. Eng.*, 103(HY7), 699–711.

Strelkoff, T. S. (1993). "Flow simulation for surface irrigation design." *Management of irrigation and drainage systems: Integrated perspectives*, R. G. Allen, ed., ASCE, Park City, UT, 899–906.

Strelkoff, T. S., and Clemmens, A. J. (2007). "Hydraulics of surface systems." *Design and operation of farm irrigation systems*, G. Hoffman and R. G. Evans, eds., American Society of Agricultural and Biological Engineers, St. Joseph, MI.

Strelkoff, T. S., Clemmens, A. J., and Bautista, E. (2009). "Field properties in surface irrigation management and design." *J. Irrig. Drain. Eng.*, 10.1061/(ASCE)IR.1943-4774.0000119, 525–536.

Strelkoff, T. S., Clemmens, A. J., and Bautista, E. (2012). "Shape factors for elements of the infiltration profile in surface irrigation: Generic approach." *J. Irrig. Drain. Eng.*, 10.1061/(ASCE)IR.1943-4774.0000413, 485–488.

Trout, T. (1992). "Furrow flow velocity effect on hydraulic roughness." *J. Irrig. Drain. Eng.*, 10.1061/(ASCE)0733-9437(1992)118:6(981), 981–987.

USDA-SCS. (1974). "Border irrigation." *National engineering handbook*, USDA, Soil Conservation Service, Washington, DC.

Walker, W. R., and Skogerboe, G. V. (1987). *Surface irrigation. Theory and practice*, Prentice Hall, Englewood Cliffs, NJ.

Zerihun, D., Furman, A., Warrick, A. W., and Sanchez, C. A. (2005). "Coupled surface-subsurface flow Model for improved basin irrigation management." *J. Irrig. Drain. Eng.*, 10.1061/(ASCE)0733-9437(2005)131:2(111), 111–128.