USGS
science for a changing world

Prepared in cooperation with
the Pikes Peak Regional Water Authority and the Colorado Water Conservation Board

Upgrades to a Fortran Program for Estimating Stream Transit Losses of Reusable Water, El Paso and Pueblo Counties, Colorado

Scientific Investigations Report 2018–5163

U.S. Department of the Interior
U.S. Geological Survey

Cover. Image showing transit-loss program code snippet, converted from FORTRAN77 to Fortran 2003.

Upgrades to a Fortran Program for Estimating Stream Transit Losses of Reusable Water, El Paso and Pueblo Counties, Colorado

By Susan J. Colarullo and Lisa D. Miller

Prepared in Cooperation with the
Pikes Peak Regional Water Authority and the
Colorado Water Conservation Board

Scientific Investigations Report 2018–5163

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
DAVID BERNHARDT, Acting Secretary

U.S. Geological Survey
James P. Reilly II, Director

U.S. Geological Survey, Reston, Virginia: 2019

Suggested citation:
Colarullo, S.J., and Miller, L.D., 2019, Upgrades to a Fortran program for estimating stream transit losses of reusable water, El Paso and Pueblo Counties, Colorado: U.S. Geological Survey Scientific Investigations Report 2018–5163, 21 p., https://doi.org/10.3133/sir20185163.

ISSN 2328-0328 (online)

Acknowledgments

The authors wish to thank Gary Krammes, who helped compile the Fortran program and provided archived daily flow output used to verify Fortran program modifications made while upgrading from FORTRAN77 to the Fortran 2003 standard.

Contents

Acknowledgments iii
Abstract 1
Introduction 1
    Purpose and Scope 3
    Description of Study Area 3
Description of the Transit-Loss Accounting Program 4
    Streambank Storage 5
    Channel Storage 6
    Evaporation 6
    Other Losses 6
    Segment and Subreach Level Transit Loss Calculations 7
    Segment Loop 7
    Subreach Loop 8
Transit-Loss Accounting Program Modifications, 1987–2012 8
Upgrade of the Transit-Loss Accounting Program from FORTRAN77 to Fortran 2003 10
    Changes to Improve Readability 11
        Comments and Continuation Lines 11
        Use of Free Format 12
        Line Indentation 12
        Replacement of Relational Operators 12
        Named IF and DO Constructs 12
        DO-EXIT and DO-CYCLE Constructs 12
        The CASE Construct 12
        Long Variable Names 13
    Changes that Protect Variables and Fixed Values 13
        Removal of Implicit Typing 13
        Conversion of Fixed-Value Variables to Parameters 13
        Incorporation of External Common Blocks 14
        Removal of Global Variables as Subroutine Arguments 14
        Use of the INTENT Attribute 14
        Use of Modules to Encapsulate Variables 14
    Changes in Array Declaration and Processing 14
        Dynamic Array Allocation 15
        Whole Array Operations 15
    Addition of Input Error Handling 15
    Verification of Upgraded Fortran Program 15
Summary 19
References Cited 20

Plate

1. Nodes and subreaches along Monument and Fountain Creeks and location of streamgaging stations, wastewater-treatment facilities, and diversions used in new transit-loss accounting program ..... available online

Figures

1. Map of study area showing Fountain and Monument Creeks and the extent of the adjoining alluvium that comprises the alluvial aquifers 2
2. Graph showing delivery node return flow predicted using upgraded Fortran 2003 program in relation to delivery node return flow predicted using original FORTRAN77 program 17
3. Graph showing difference in predicted delivery node return flows made using upgraded Fortran 2003 program and original FORTRAN77 program relative to delivery node return flows predicted using original FORTRAN77 program 17
4. Graph showing subreach return flow predicted using upgraded Fortran 2003 program in relation to subreach return flow predicted using original FORTRAN77 program in Monument Creek 17
5. Graph showing difference in predicted subreach return flows made using upgraded Fortran 2003 program and original FORTRAN77 program relative to subreach return flows predicted using original FORTRAN77 program in Monument Creek 18
6. Graph showing subreach return flow predicted using upgraded Fortran 2003 program in relation to subreach return flow predicted using original FORTRAN77 program in Fountain Creek 18
7. Graph showing difference in predicted subreach return flows made using upgraded Fortran 2003 program and original FORTRAN77 program relative to subreach return flows predicted using original FORTRAN77 program in Fountain Creek 18

Tables

1. Streamgaging stations along Monument and Fountain Creeks and at selected tributaries used in transit-loss accounting program 4

Conversion Factors

U.S. customary units to International System of Units

Multiply	By	To obtain
	Length	
inch (in.)	2.54	centimeter (cm)
inch (in.)	25.4	millimeter (mm)
foot (ft)	0.3048	meter (m)
mile (mi)	1.609	kilometer (km)
	Area	
square mile (mi^2)	259.0	hectare (ha)
square mile (mi^2)	2.590	square kilometer (km^2)
	Volume	
cubic foot (ft^3)	0.02832	cubic meter (m^3)
acre-foot (acre-ft)	1,233	cubic meter (m^3)
	Flow rate	
cubic foot per second (ft^3/s)	0.02832	cubic meter per second (m^3/s)
	Hydraulic conductivity	
foot per day (ft/d)	0.3048	meter per day (m/d)

Vertical coordinate information is referenced to the North American Vertical Datum of 1988 (NAVD 88).

Horizontal coordinate information is referenced to the North American Datum of 1983 (NAD 83).

Elevation, as used in this report, refers to distance above the vertical datum.

Abbreviations

CSU	Colorado Springs Utilities
DWR	Division of Water Resources
RRF	Reusable return flows
SECWCD	Southeastern Colorado Water Conservancy District
USGS	U.S. Geological Survey
WWTF	Wastewater-treatment facility

Upgrades to a Fortran Program for Estimating Stream Transit Losses of Reusable Water, El Paso and Pueblo Counties, Colorado

By Susan J. Colarullo and Lisa D. Miller

Abstract

In 2016, the U.S. Geological Survey, in cooperation with Pikes Peak Regional Water Authority and the Colorado Water Conservation Board, began a study to modernize a Fortran transit-loss accounting program developed by the U.S. Geological Survey to estimate net reusable flows in Fountain and Monument Creeks in El Paso and Pueblo Counties, Colorado. More than 6,000 lines of this FORTRAN77 transit-loss accounting program were revised to comply with the newer Fortran 2003 standard. The upgrade to the newer standard involved making changes in formatting and syntax on each line and, when available, adding new programming constructs that comply with the new standard. These upgrades produced a more readable Fortran program that includes safeguards to prevent accidental mistyping of variables and unintentional changes in named constants during program execution. Program revisions also introduced dynamic array allocation, whole array processing, and handling of input errors to the upgraded transit-loss accounting program.

During the upgrade from FORTRAN77 to the Fortran 2003 standard, revisions were made incrementally to the original transit-loss Fortran program. Because FORTRAN77 is a subset of Fortran 2003, the legacy FORTRAN77 statements and the upgraded Fortran 2003 statements can be compiled within the same program, permitting program revisions to be gradually phased in on a line-by-line basis. This incremental approach helped mitigate risks of introducing logic errors into the Fortran program that could produce incorrect transit-loss estimates.

Verification of the upgraded transit-loss accounting program focused on reproducing archived reusable return flows (RRF) for historical daily runs from January 5, 2015, to October 31, 2018. Because interim files storing daily streambank losses were not historically archived, no record of antecedent streambank storage losses to hydraulically connected alluvial deposits were available to provide initial conditions for each daily run. To overcome the problem of missing historical bank storage and recovery files that contain important information relating to antecedent streambank storage conditions, a 104-day "spin-up" period was required before RRFs calculated by the upgraded program and the original program matched. Estimated daily reusable return flows from archived output generated by the original program and output generated by the upgraded program were compared after this initial "spin-up" period. Daily reusable return flow estimates at delivery nodes and at the bottoms of subreaches from the upgraded Fortran program matched those output by the original program to within 0.01 and 0.0001 cubic feet per second, respectively.

Introduction

Like other cities in eastern Colorado, Colorado Springs supplements its water supply with non-native (transmountain) water. Under Colorado water law, waters not native to a watershed may be claimed for reuse as long as they can be distinguished from native streamflows. Non-native water imported from outside a watershed can be used and reused to the point of extinction if the amount of imported water can be quantified at its points of intended reuse (Radosevich and others, 1976). To more effectively manage this reusable water, a study was completed by the U.S. Geological Survey (USGS) in 1987, in cooperation with Colorado Springs Utilities (CSU), to develop a physically based methodology able to distinguish native (waters naturally occurring in a watershed) from non-native waters (water imported into the watershed from another watershed). The methodology, developed by Kuhn (1988), also accounts for temporary streambank losses to alluvial aquifers that are hydraulically connected to Fountain and Monument Creeks in El Paso and Pueblo Counties, temporary losses to channel storage, and permanent evaporation losses from the stream surface (fig. 1). In its original inception, the transit-loss methodology was incorporated into an accounting computer program that calculates the total amount of daily reusable return flow (RRF) reaching the Arkansas River by quantifying and tracking non-native RRFs in Fountain Creek from the Las Vegas Street Wastewater-Treatment Facility (WWTF) to its confluence with the Arkansas River in Pueblo, Colo. (fig. 1).

The transit-loss accounting program, which has been in continuous daily use in some form since April 1989, has provided water-rights administrators with a tool to effectively

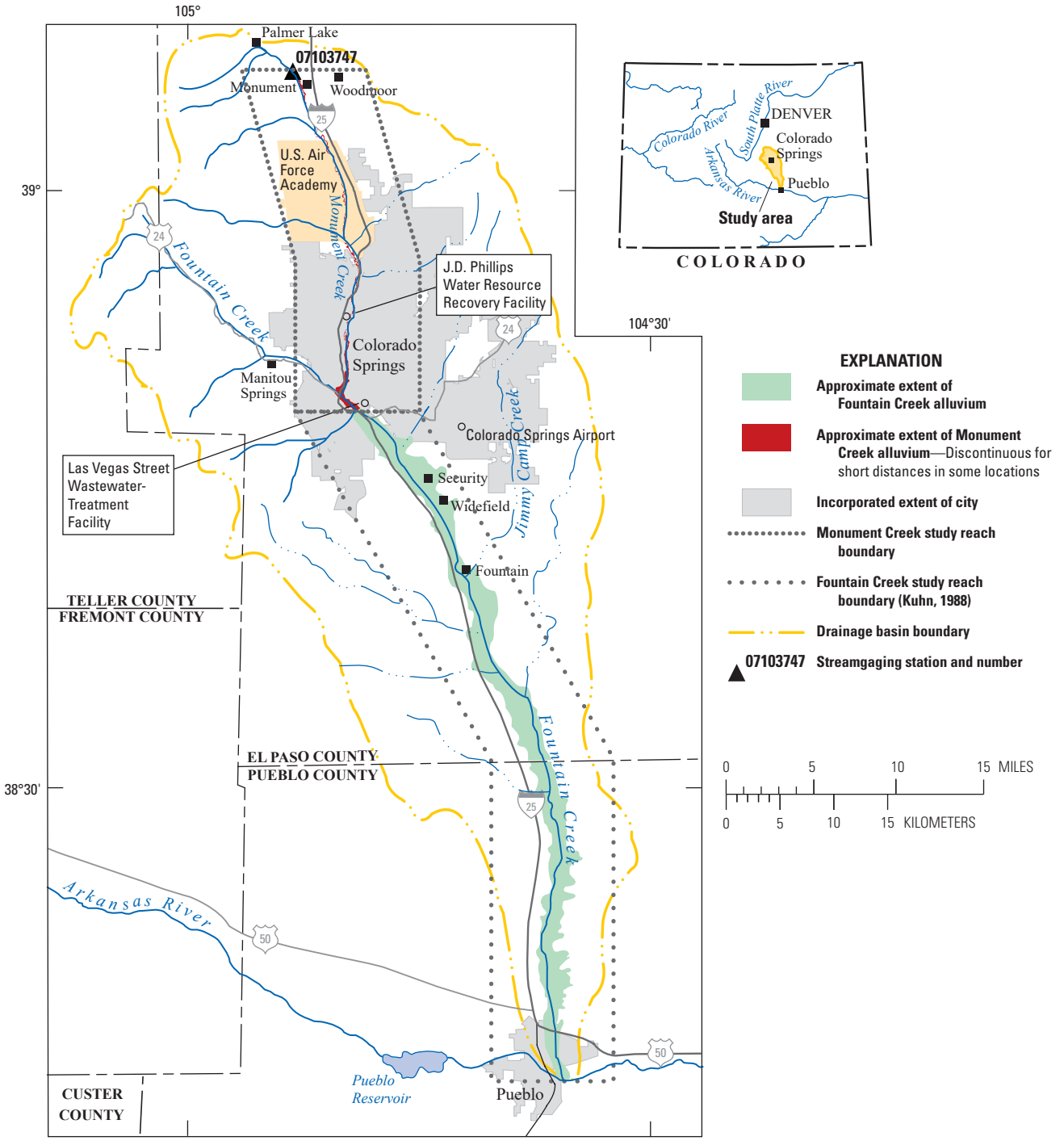

Figure 1. Study area showing Fountain and Monument Creeks and the extent of the adjoining alluvium that comprises the alluvial aquifers.

manage RRFs and administer water diversion priorities along Fountain Creek on a daily basis. Over time, this transit-loss accounting program has been expanded to include Monument Creek, a tributary of Fountain Creek, and incorporate and track more RRFs from other entities in the watershed (Kuhn and Arnold, 2006). The total volume of RRF flowing into Monument and Fountain Creeks and tracked with the transit-loss accounting program varies from year to year but has generally increased over time as the population in the watershed has grown. With increased urbanization, greater quantities of water must be imported from outside the watershed in order to sustain the growing population. For example, the volume of RRF input and tracked using the transit-loss accounting program was 27,785 acre-feet from August 2006 to July 2007 and 39,307 acre-feet from August 2016 to July 2017, an increase of about 41 percent. This trend towards increasing RRFs underscores the importance of using a physically based methodology to quantify the net volume of RRFs available to fulfill water needs in an area experiencing substantial growth.

The current transit-loss accounting program is written in FORTRAN 77 and is run daily by Colorado Division of Water Resources (DWR) District 10 Water Commissioner to administer water rights in the Fountain Creek watershed. Program updates are needed periodically to make modifications and maintain compatibility with evolving computer software and hardware. To modernize the transit-loss accounting program and ensure compatibility with evolving software and hardware, the USGS, in cooperation with Pikes Peak Regional Water Authority and the Colorado Water Conservation Board, began an effort in 2016 to upgrade the existing FORTRAN77 transit-loss program so that it complies with more recent Fortran 2003 programming standards.

Purpose and Scope

The purpose of this report is to describe updates and structural changes made during 2016–18 to the existing FORTRAN77 transit-loss accounting program source code so that it adheres to more recent Fortran 2003 programming standards. The report documents changes that have been made to the program over the past 30 years (1989–2018) and provides a description of the manner in which the transit-loss accounting program computes transit losses for reusable waters along Fountain and Monument Creeks in eastern Colorado. The upgraded Fortran 2003 program described in this report is available at https://code.usgs.gov/water/Transit-Loss.

Description of Study Area

The Fountain Creek watershed drains an area of about 930 square miles and encompasses the cities of Palmer Lake, Monument, Colorado Springs, Security, Widefield, Fountain, and parts of Pueblo (fig. 1). The study area spans the alluvial valley from USGS streamgaging station 07103747 Monument Creek at Palmer Lake to the confluence of Fountain Creek and the Arkansas River. The area is 78.3 river miles (mi) long and varies in width from about 100 feet (ft) to 2 miles (mi), depending on the local width of hydraulically connected alluvial streambanks that temporarily store and attenuate native and reusable flows. Elevation in the study area decreases from roughly 7,300 ft, referenced to the North American Vertical Datum of 1988, at Palmer Lake to 4,650 ft at the confluence of Fountain Creek and the Arkansas River. Climate in the study area is semiarid. Average annual precipitation decreases as elevation decreases, ranging from 21.8 inches at Palmer Lake to 11.8 inches at Pueblo (Western Regional Climate Center, 2018). Mean monthly temperatures increase to the south with decreasing elevation.

Streamflow in Fountain Creek and Monument Creek originates from (1) snowmelt runoff from mountainous areas in the headwaters generally during April, May, and June; (2) rainfall runoff from thunderstorms, primarily from May through September; (3) return flow from municipal, agricultural, domestic, and industrial water uses; and (4) groundwater inflow from alluvial aquifers (Livingston and others, 1976). Streamflow is highly variable from day to day, particularly during summer months when runoff from frequent thunderstorms can cause rapid changes in streamflow. In Monument Creek, streamflow generally increases in the downstream direction because accumulation of native flows more than compensates for losses caused by surface-water diversions. Unlike Monument Creek, streamflow along Fountain Creek does not consistently increase in the downstream direction because large diversions from Fountain Creek at multiple locations, particularly during spring and summer, are not offset by increases in native flows. Most of the surface-water diversions from both creeks are used for irrigation of crops in the valley and to support the needs of local municipalities, with small amounts of diverted streamflow used to artificially recharge the alluvium.

Because Fountain and Monument Creeks are hydraulically connected to the alluvium at locations where the alluvial fill is laterally continuous, there is substantial ongoing interaction between surface water flowing through both creeks and groundwater in adjoining, hydraulically connected alluvial aquifers (Livingston and others, 1976). The Quaternary alluvium adjacent to Fountain Creek is composed of gravel and sand with lesser quantities of silt and clay, underlain by the relatively impermeable Cretaceous Pierre Shale. Alluvium adjacent to Fountain Creek ranges in width from about 0.75 to 2.0 mi and in thickness from a few feet to about 100 ft (Jenkins, 1964; Cain and Edelmann, 1986). Hydraulic conductivity in the alluvial aquifer along Fountain Creek ranges from about 400 to 1,600 feet per day (ft/d) (Jenkins, 1964; Wilson, 1965; Taylor, 1975). Stream-channel and flood-plain alluvium along Monument Creek is composed of Holocene-age sand, gravel, silt, and clay with lesser amounts of pebbles and cobbles. Unlike the alluvial deposits adjacent to Fountain Creek, Monument Creek alluvium ranges in width from 0 to 2,000 ft and is generally less than 10 ft thick. Hydraulic conductivity of the Monument Creek alluvium is estimated to range from

about 100 to 1,000 ft/d with an average value of about 400 ft/d (Kuhn and Arnold, 2006). Because stream-channel and floodplain deposits along Monument Creek generally are thin and groundwater storage in the adjacent alluvium is small, the Monument Creek alluvial aquifer is not used extensively for groundwater supply.

Description of the Transit-Loss Accounting Program

Complex natural processes that depend on a number of factors must be modeled in order to simulate the additions and subtractions of transported water, including storage of reusable surface waters in the hydraulically connected alluvial aquifer, the velocity of the water wave induced by upstream releases of reusable waters, changes in the proportion of imported water relative to native water, and transfer of water from stream surfaces into the atmosphere via evaporation. Transit losses can be accurately predicted by the transit-loss accounting program using output from a numerical stream-aquifer model that explicitly accounts for natural conveyance losses that occur by three distinct mechanisms: (1) temporary losses to alluvial streambank storage, (2) temporary losses to in-channel storage, and (3) permanent losses caused by evaporation from the stream surface. Potential transit losses from groundwater withdrawals, specific retention in the alluvial aquifer, evaporation from soil surfaces, transpiration from riparian phreatophytes growing in soils overlying the alluvial aquifer, and inadvertent diversions were considered insignificant and were not included in the numerical model used to quantify net reusable flows (Kuhn, 1988; Kuhn and Arnold, 2006).

To build the stream-aquifer model, study reaches along Fountain and Monument Creeks were divided into stream segments, nodes, and subreaches. Stream segments were defined as the parts of the streams that are between main stem streamgaging stations (table 1). Plate 1, which is reproduced from Kuhn and others (2007), shows locations of streamgaging stations, wastewater-treatment facilities, and ditch diversion points along Monument and Fountain Creeks and illustrates the relation between nodes and subreaches used by the stream-aquifer model. Thirty-four nodes were established at or near points along the streams where inflows or outflows occur, such as at wastewater-treatment facility discharge locations and streamflow diversion points (pl. 1). These nodes delimit the 33 subreaches (pl. 1), which represent subdivisions within stream segments that are assumed to have uniform hydraulic and hydrologic characteristics (Kuhn, 1988; Kuhn and Arnold, 2006).

Table 1. Streamgaging stations along Monument and Fountain Creeks and at selected tributaries used in transit-loss accounting program.

[--, not determined; USGS, U.S. Geological Survey; USGS station numbers shown on plate 1]

USGS station number	Site type	Station name	Drainage area (square miles)
07103747	Main stem	Monument Creek at Palmer Lake	25.7
07103755	Main stem	Monument Creek below Monument Lake near Monument	30.2
07103780	Main stem	Monument Creek above Northgate Boulevard at U.S. Air Force Academy	81.4
07103800	Tributary	West Monument Creek at U.S. Air Force Academy	14.8
07103990	Tributary	Cottonwood Creek at mouth at Pikeview	18.9
07104000	Main stem	Monument Creek at Pikeview	203
07104905	Main stem	Monument Creek at Bijou Street at Colorado Springs	235
07105500	Main stem	Fountain Creek at Colorado Springs	392
07105530	Main stem	Fountain Creek below Janitell Road below Colorado Springs	413
07105600	Tributary	Sand Creek above mouth at Colorado Springs	52.4
07105800	Main stem	Fountain Creek at Security	500
07105900	Tributary	Jimmy Camp Creek at Fountain	65.4
07106000	Main stem	Fountain Creek near Fountain	672
[1]07106330	Main stem	Fountain Creek near Piñon	865
07106500	Main stem	Fountain Creek at Pueblo	925
[2]FOUMOCO	Main stem	Fountain Creek at mouth	--

[1]U.S. Geological Survey station number changed from 07106300 to 07106330 on October 1, 2018.

[2]Station operated by Colorado Division of Water Resources; station not used in transit-loss accounting program, but is used by Division Engineer in administration of reusable water rights.

It is important to note the distinction between the stream-aquifer model and the transit-loss accounting program used to estimate RRFs at downstream points of use. The stream-aquifer model was used to estimate streambank-storage and channel-storage losses for numerous combinations of native and reusable flows within each subreach. The transit-loss accounting program uses these streambank-storage and channel-storage losses estimated by the stream-aquifer model to estimate net transit losses along Monument and Fountain creeks. Permanent evaporation losses from reusable waters, a small but critical loss component that can have important hydrologic implications during periods of low flow, are computed directly by the transit-loss program.

The stream-aquifer model required input of physical characteristics, and aquifer and channel hydraulic properties, for each stream subreach. Required input characteristics and properties include length, width, transmissivity, and storage coefficient of the alluvial aquifer, as well as length, slope, wave celerity, and the wave-dispersion coefficient of the stream channel. Transmissivity is a measure of the rate at which water can flow through an aquifer and is equal to the product of hydraulic conductivity and saturated thickness, whereas the storage coefficient is defined as the volume of water that an aquifer can take into or release from storage per unit change in hydraulic head and is approximately equal to the specific yield for an unconfined aquifer such as the Fountain Creek and Monument Creek alluvium (Kuhn and Arnold, 2006). Wave celerity provides a measure of the velocity of a water wave through a stream reach, whereas the wave-dispersion coefficient describes the amount of "spread" or attenuation of the water wave as it moves through a stream reach (Kuhn and Arnold, 2006).

For modeling purposes, to approximate releases of RRFs, recorded streamflow hydrographs were selected that exhibited a period of constant streamflow followed by a pronounced increase in streamflow and a subsequent return to constant streamflow (Kuhn, 1988; Kuhn and Arnold, 2006). Calibration of the stream-aquifer model involved manual changes in poorly known values of aquifer transmissivity and storage coefficient, along with wave celerity and wave dispersion coefficient, until the sum of squared differences between simulated streamflow and recorded streamflow were minimized. Known physical characteristics of the aquifer and channel, including aquifer width and length, and channel length and slope, were not varied during model calibration. Verification of the stream-aquifer model involved running a number of simulations for stream segments along both creeks bounded by streamgaging stations, for which concurrent, continuous hydrographs were available, and comparing simulated and recorded streamflow values to ensure that the adjusted model parameters produced simulated hydrographs that closely matched recorded hydrographs.

Streambank Storage

Owing to their ability to change the timing and magnitude of flows and sustain base flows during periods of low flow, bank storage losses can have important hydrologic implications. Responsible administration of water rights requires that RRF streambank losses be accurately quantified using the physically based stream-aquifer model. To simulate losses of RRFs to alluvial streambank storage, the stream-aquifer model treats each upstream release of RRF as a wave preceded by uniform antecedent native streamflow. During passage of each wave, hydraulic head in the stream increases relative to head in the adjacent hydraulically connected alluvium, creating an outward hydraulic gradient that pushes channel water into streambanks as the wave of released RRF passes. The percent RRF lost to streambank storage depends on the relative amounts of reusable and native flow, with larger streambank storage losses associated with greater quantities of RRF relative to native flow. Streambank-storage losses associated with a given wave of released RRF are assumed to occur only on the day of release.

Between successive waves of released reusable water, when stream stage typically falls below the hydraulic head in the adjacent alluvium and the hydraulic gradient is reversed, most of the reusable water held in streambank storage returns to the stream channel as base-flow contributions during intervening periods of recovery. However, recovery of streambank storage losses occurs at a much slower rate than the rate at which the water enters into streambank storage. This rate of recovery is assumed to decay exponentially over time. The exponentially decaying rate of recovery between successive waves is generally so slow that consideration of the extremely long recovery periods needed to recover all streambank-storage losses is impractical from a computational perspective (Kuhn, 1988). As a consequence, a small quantity of RRF is assumed to be permanently lost to streambank storage during execution of the transit-loss accounting program.

The recovery period required for all streambank-storage water lost from a given water wave to be regained by the stream varies with the length of the stream channel, the magnitude of the RRF release, and the hydrologic properties of the alluvial streambank. Because streambank storage represents the single largest component of transit losses for reusable waters (Kuhn, 1988), streambank losses can have a pronounced effect on the timing and magnitude of RRF delivery to downstream points of use. In the highly permeable, hydraulically connected alluvial sediments adjacent to Fountain Creek, Kuhn (1988) reported that streambank-storage loss can vary from about 50 percent of the RRF on the day of release to about 2 percent after a 180-day recovery period.

Fountain Creek subreaches were assumed to recover streambank-storage water more slowly than Monument Creek

subreaches because the alluvial aquifer adjacent to Fountain Creek is thicker and more laterally extensive. Kuhn (1988) used a uniform recovery period of 180 days to determine the relations between percentage of streambank-storage loss remaining in storage and the recovery-period day for subreaches along Fountain Creek. However, when the program was implemented, a uniform 60-day recovery period for Fountain Creek was agreed upon by the Colorado Department of Water Resources Division 2 Engineer and CSU (Kuhn and Arnold, 2006). On the basis of results from the stream-aquifer model, the 60-day recovery period corresponds to the length of time required for roughly 91.6 percent of streambank-storage losses in Fountain Creek subreaches to be recovered. Similarly, the Colorado Department of Water Resources Division 2 Engineer and cooperating water-management entities administratively accepted that accounting of gains from streambank storage for Monument Creek subreaches would be truncated at 91.6-percent recovery. The resulting Monument Creek subreach recovery periods varied from 1 to 28 days (Kuhn and Arnold, 2006). The practice of intentionally underestimating the amount of reusable water recovered from streambank storage by allowing 8.4 percent of reusable water to remain in streambank storage helps mitigate the risk that existing water rights could be compromised owing to inaccuracies in the transit-loss accounting methodology.

Channel Storage

The same calibrated stream-aquifer model used to estimate streambank-storage losses was used to determine channel-storage losses that occur during downstream conveyance of RRFs. Transit losses attributed to channel storage occur as the wave associated with a release of reusable water passes through a subreach (Kuhn, 1988; Kuhn and Arnold, 2006). As the wave passes, channel storage increases. After passage of the water wave, channel storage rapidly diminishes as it becomes a component of flow further downstream.

To determine whether substantial channel-storage losses extend beyond the day of reusable water release, test simulations were performed using the channel-routing component of the stream-aquifer model. These simulations indicated that, owing to the short length of the subreaches relative to the magnitude of wave displacement over a 1-day period, in-channel storage losses from RRFs can be assumed to be fully recovered within 1 day of release (Kuhn, 1988; Kuhn and Arnold, 2006). Model simulations also indicated that channel-storage losses in most subreaches on the day of release amount to only about 10 percent of RRF. In long subreaches, model simulations indicated that channel-storage losses on the day of release account for about 20–30 percent of RRFs (Kuhn, 1988; Kuhn and Arnold, 2006). These percentages were used by the accounting program to estimate channel-storage losses on the day of release, with all losses recovered the next day. Because channel-storage losses on any given day are fully recovered in the form of gains from channel storage after passage of the water wave, these losses are considered temporary. However, they do affect the timing and amount of reusable water delivery at downstream points of use.

Evaporation

Evaporation loss for transport of RRFs was determined using methods described by Livingston (1978) because the stream-aquifer model used to determine streambank storage and channel storage contained no provision for evaporation. Evaporation losses occur either by direct evaporation from the stream surface or by indirect evaporation of water stored in alluvial streambanks. Daily evaporation losses from the stream surface depend on the time of year but generally amount to less than 5 percent of daily RRF releases (Kuhn, 1988). Unlike temporary losses associated with streambank and channel storage, evaporation losses were considered permanent.

Daily evaporation data were not readily available for stream surfaces and had to be estimated. For Fountain Creek, the rate of evaporation from the stream surface was estimated on the basis of historical monthly pan-evaporation data collected at Pueblo City Reservoir for the 1941–68 period (Kuhn, 1988). For Monument Creek, monthly pan-evaporation data for 2006 estimated by the National Weather Service at the Colorado Springs airport were used to estimate the rate of daily evaporation (Kuhn and Arnold, 2006). Monthly evaporation data were adjusted by a pan coefficient of 0.72 to determine daily evaporation (Farnsworth and others, 1982).

The rate of volumetric evaporative loss was estimated by multiplying the evaporation rate by subreach length and incremental stream-width increase resulting from the RRF release. Stream widths were estimated using a single linear regression relation between stream width and streamflow derived using data pairs collected at streamgaging stations in the stream. Daily evaporation loss was estimated after losses to and gains from streambank storage and channel storage were determined. On average, daily evaporative loss in a subreach has been shown to be small relative to streambank and channel storage losses (Kuhn and Arnold, 2006).

Other Losses

Additional sources of permanent transit losses include groundwater withdrawals, specific retention in the alluvial aquifer, transpiration, and inadvertent diversion. Streamflow transit losses induced by groundwater withdrawals occur when groundwater withdrawals lower hydraulic head in the alluvial aquifer adjacent to the stream to below stream stage, causing more reusable waters flowing through the stream channel to be pulled into the streambank than would occur in the absence of pumping. However, because groundwater appropriations are determined on the basis of native streamflow quantities, groundwater withdrawals were assumed to be derived strictly from the native streamflow component and to have no effect on streambank losses of imported waters.

Loss of water to specific retention, or water retained by surface tension in a porous medium against the pull of gravity, is a one-time irreversible loss that was not considered substantial for this study. Transit losses resulting from transpiration of vegetation along alluvial banks also were not included in the stream-aquifer model. Some of these losses would, to some extent, be derived from the small fraction of RRFs assumed to be permanently lost to streambank storage via specific retention. Stream-stage observations along Fountain Creek indicate that the maximum increase in stream stage caused by an RRF release is unlikely to exceed 0.5 ft at any given time (Kuhn, 1988). Because decreased depth to water in the adjacent alluvial aquifer caused by a 0.5-ft increase in stream stage is expected to be substantially smaller than 0.5 ft, it was assumed that any reduction in depth to water in the alluvial aquifer is not likely to enhance the rate of transpiration.

Inadvertent diversion caused by accidental overtopping of diversion ditches during periods of high stream stage also was not considered because established water rights are based on native streamflows, not RRFs. Accidental overtopping was assumed to represent a loss to native waters only and would not diminish the quantity of reusable water (Kuhn, 1988).

Segment and Subreach Level Transit Loss Calculations

Upstream releases of reusable return flows to streams can be viewed as intermittent pulses of water that are conveyed downstream along with native streamflow components. When these pulses of imported water are large relative to native flows, they act much like high-magnitude "spikes" that can push substantial quantities of channel water into adjacent alluvial streambanks under steep outward gradients. In contrast, small upstream releases of reusable flows are generally swamped by native flows, manifesting as low-magnitude waves that push insignificant amounts of water into streambank storage. The quantity of imported water attenuated by streambanks on any given day is dictated by this complex interplay between native flows and streambank losses from reusable flows. Kuhn (1988) resolved this dependency between native and net reusable flows by keeping native and RRF flow calculations separate but allowing them to exchange information using an iterative approach.

The iterative calculations are made using the system of nodes, subreaches, and segments previously developed by Kuhn (1988) and Kuhn and Arnold (2006) that are shown on plate 1. This system allows estimated bank storage losses caused by RRF releases along Fountain and Monument Creeks to be conditioned on total streamflow observed at streamgaging stations located at the ends of the stream segments but removes inherent dependencies between native flows and streambank losses caused by intermittent releases of reusable flows by adding smaller-scale subreaches. These subreaches enable a separate but parallel set of calculations to determine net bank storage losses from imported waters.

Subreaches represent the basic stream-channel element used in the stream-aquifer model. All physical properties of the stream, including channel length and slope, aquifer length and width, and wave celerity and wave dispersion coefficient, are defined at the scale of this smallest stream-channel element. It is at this subreach level where streambank storage losses are calculated, conditioned on native streamflows estimated at the segment level. Once subreach transit losses for RRFs are estimated for all subreaches contained in the segment, these losses are communicated back to the segment level, where they are used to refine previous estimates of the native streamflow component of total gaged streamflow.

Iterative passing of estimated native streamflows from each segment to its subreaches, followed by passing of estimated streambank storage losses from the subreaches back to the segment, continues until the estimated change in total streamflow between successive iterations falls below the specified closure tolerance of 0.5 percent. After convergence occurs for the current segment, the transit-loss model continues to the next downstream segment. The segment calculations, along with their subreach calculations, are repeated until the convergence criterion is met for every segment. At this point, all dependencies between native streamflow estimates and estimates of transit losses become fully resolved, temporary estimates of transit losses and net RRFs are finalized, and all calculated native and reusable flow components are output to text files.

Segment Loop

Changes in native streamflow between streamgaging stations are estimated at the stream-segment level. Dependencies between native streamflow and losses from RRFs are resolved using an iterative approach that transfers estimates of native streamflows and RRF losses between a stream segment and all subreaches contained within the segment. From a programming perspective, these two levels of flow estimation are easily implemented by nesting a subreach loop within a segment loop. Changes in native streamflows estimated in the outer segment loop are passed as "known" values to the inner subreach loop, where streambank-storage-loss calculations are made. These streambank-loss values are then passed back to the outer loop as "known" variables for the next iteration. Iteration continues until very little change in estimated native streamflows or estimated transit losses occurs between successive iterations.

Conditional losses or gains in native streamflow within a stream segment are estimated by computing the difference between total streamflow measured at the upstream and downstream ends of the segment, adjusted by adding known tributary inflows and known releases of RRFs and subtracting known diversions of native streamflow and RRFs. Because all components of the mass balance must be known, unknown streambank-storage losses in the segment loop are initially assigned values of zero. Because data related to

precise locations where gains or losses in native streamflow occur are unavailable, these gains or losses are assumed to be uniformly distributed along the segment during subsequent subreach computations.

Variables used to estimate the change in native streamflow along the segment per stream mile include the following known quantities: total gaged streamflow at the upstream node, total gaged streamflow at the downstream node, total RRF at the downstream node of the previous segment, total RRF releases within the segment, total reusable and native streamflow diversions within the segment, and total channel length of the segment. The general sequence of operations used to estimate total native water loss or gain per stream mile in the current segment is shown in Kuhn and others (2007, fig. 6).

Subreach Loop

It is at the subreach level that the accounting program estimates losses from reusable flows. Segment-level estimates of changes in native streamflow provide the basis for estimating transit losses for RRFs to streambank storage, channel storage, and evaporation in these subreach-level computations. Estimates of changes in native streamflow in a segment are based on the assumption that these changes occur uniformly across the segment, with the same changes in native streamflows at the upstream and downstream ends of the segment. As a consequence of this assumption, local changes in native streamflows caused by the mid-segment influx of native water from tributaries, native streamflow diversions, or native streambank-storage losses and gains will not be reflected in subreach calculations.

To account for the effects of changes in native streamflows across a subreach on predictions of RRF streambank-storage losses, bank-storage loss is multiplied by an adjustment factor. This adjustment factor is estimated by running the stream-aquifer model for a set of ratios of upstream to downstream native streamflows and predicting streambank-storage losses of reusable water for each subreach. These RRF streambank-storage loss predictions are then divided by predicted bank-storage losses that would occur if native streamflows on the upstream and downstream ends of the subreach are equal. The resulting set of bank-storage loss adjustment factors, which typically range from 0.80 to 1.20, was then regressed against the ratios of upstream to downstream native streamflows using nonlinear regression, and the regression slope was used to scale bank-storage losses of RRFs up or down, depending on whether downstream native streamflow is smaller or larger than upstream native streamflow (Kuhn and Arnold, 2006).

Once the change in native streamflow per mile for a segment is passed to the inner subreach loop, unknown values related to all non-native streamflow components for the subreaches spanning the current segment are estimated. These include subreach estimates of RRF, streambank storage losses and gains, channel-storage losses and gains, and evaporation losses. All estimates are based on known variables from the current iteration of the segment loop, from the previous iteration of the subreach loop, or from the previous iteration of the subreach loop corresponding to the immediately upstream subreach. The general sequence of operations used to estimate unknown variables for a subreach is shown in Kuhn and others (2007, fig. 7).

Transit-Loss Accounting Program Modifications, 1987–2012

Prior to development and implementation of the USGS transit-loss accounting program in 1989, exchanges of RRFs discharged into Fountain Creek at the CSU's Las Vegas Street WWTF, formerly known as the Colorado Springs WWTF, were made in accordance with an interim exchange agreement. As designated by the interim agreement, no exchange of RRF could be made to fulfill downstream water rights if streamflow was not continuous in Fountain Creek. If Fountain Creek flowed continuously during the irrigation season, transit losses were assessed at 0.07 percent per mile if the flow at streamgaging station 07106300 (Fountain Creek near Pinon) recorded a flow of 500 cubic feet per second (ft^3/s) or greater. If flow at station 07106300 was less than 500 ft^3/s, transit losses were assessed at 10 percent. During the non-irrigation season, transit losses were assessed at 0.07 percent per mile (Kuhn, 1988).

In 1987, the USGS, in cooperation with CSU, completed a study to develop a physically based methodology to assess transit losses from RRF discharged into Fountain Creek at the Las Vegas Street WWTF. The goal of the study was to replace empirical transit-loss estimates, established by the interim agreement, with a water-rights management tool that estimates transit losses using a model that more accurately accounts for natural processes known to deplete flow in Fountain Creek during downstream transit (Kuhn, 1988). The study involved (1) calibration and verification of the USGS J349 stream-aquifer model that explicitly accounts for bank-storage and other transit losses, (2) use of the model to accurately predict transit losses, and (3) development of a transit-loss accounting computer program that uses transit-loss predictions output by the stream-aquifer model to calculate daily transit losses (Kuhn, 1988).

To account for dependencies between native and reusable flows known to occur during streambank loss calculations, the calibrated model was repeatedly executed to predict bank-storage losses for various combinations of RRFs and native streamflows. On the basis of these simulations, bank-storage losses were estimated for every Fountain and Monument Creek subreach for discrete combinations of RRF values ranging from 1 to 200 ft^3/s and native streamflows ranging from 0 to 1,000 ft^3/s. These loss estimates were stored in static lookup

tables used by the transit-loss accounting program to interpolate streambank storage losses for arbitrary values of RRFs and native streamflows.

Over time, modifications have been made to the transit-loss accounting program to enhance the original code and add capabilities to account for additional inflows and diversions along Fountain Creek and Monument Creek, including development of an interface to facilitate user input. Modifications made to the transit-loss accounting program during 1991–92 and 1994–95 are documented in Kuhn and others (1998). The 1991–92 modifications included modularization and enhancement of the original code and added capabilities to account for new diversions of RRFs and release of RRFs at locations along Fountain Creek other than at the Las Vegas Street WWTF. During the 1994–95 program modification, an additional streamgaging station (07105530), which is about 1 mi downstream from the Las Vegas Street WWTF, was incorporated into the program to improve the accuracy of transit losses calculations. To incorporate the new streamgaging station, an additional subreach, node, and stream segment were created (Kuhn and others, 1998). Other modifications involved adding capabilities to the model to track Fryingpan-Arkansas Project (Southeastern Colorado Water Conservancy District, 2018) reusable waters separately from other RRFs.

A study was conducted from 2004 to 2007, in cooperation with CSU, the Colorado Water Conservation Board, and the El Paso County Water Authority, to expand the Fountain Creek transit-loss accounting program by including Monument Creek, a tributary of Fountain Creek (fig. 1; Kuhn and others, 2007). Monument Creek was incorporated into the stream-aquifer model and the transit-loss accounting program in anticipation of tracking future releases of imported water from the J.D. Phillips Water Resource Recovery Facility (WRRF) and other entities in the Monument Creek watershed (Kuhn and others, 2007). The J.D. Philips WRRF, built to accommodate the greater amount of wastewater-treatment capacity needed to support the rapidly increasing population of Colorado Springs, came online in 2007. The Monument Creek study reach (pl. 1) extends from streamgaging station 07103747 (Monument Creek at Palmer Lake) downstream to the Las Vegas Street WWTF, which is about 2 miles downstream on Fountain Creek from streamgaging station 07103747 and represents the most upstream point for the original Fountain Creek transit-loss study (Kuhn, 1988).

The 2004–07 study involved (1) applying the stream-aquifer model to Monument Creek, (2) using results of the model to develop a transit-loss accounting program for Monument Creek, (3) revising the existing Fountain Creek transit-loss accounting program to accommodate ongoing and future changes in management of native and reusable streamflows, and (4) integrating the two accounting programs into a single program with a web-based interface to allow manual and automated data entry (Kuhn and Arnold, 2006; Kuhn and others, 2007). The existing Fountain Creek transit-loss accounting program was used as a prototype for the new Monument Creek transit-loss accounting program. The two study reaches (Fountain Creek and Monument Creek) were integrated into one system with 34 nodes and 33 subreaches, numbered consecutively from upstream to downstream. Nodes were established at or near points along the streams where inflows or outflows occur, such as at WWTF discharge locations and streamflow diversion points. The nodes delimit the subreaches, which are subdivisions within the study reaches that were assumed to have uniform hydraulic and hydrologic characteristics (Kuhn and others, 1988; Kuhn and Arnold, 2006). The transit-loss accounting program was modified during this study period to enable accounting and transit-loss computations for any number of RRFs, user-specified release and delivery nodes, diversion of all or a part of a RRF at any node, and accounting and transit-loss computations for augmentation flows (flows used to replace depletions from alluvial wells).

Minor modifications were made to the transit-loss accounting program between 2007 and 2008. In September 2007, the transit-loss accounting program was modified to accommodate input of augmentation flows less than 0.01 ft^3/s; allowable input increased from two decimal places to four decimal places. Code also was added to generate a comma separated value format output file, with all header and formatting stripped, so that program output could be directly uploaded by DWR. Historic transit-loss accounting program output can be downloaded from Colorado Division of Water Resources' imaged documents by searching "Fountain Creek transit loss" at https://dnrweblink.state.co.us/dwr/search.aspx?dbid=0. The number of lines available for input of RRFs was increased from 100 to 200 in April 2008 to accommodate an increasing number of RRF releases.

During 2009, changes were made to the transit-loss accounting program to enable computations of discrete streambank-storage losses and gains for each RRF. This capability was present in the original code prior to the 2007 update. However, starting with the 2007 update, all RRFs were summed to allow for the increasing and variable number of RRFs input into the transit-loss accounting program on a daily basis. To determine daily streambank-storage losses and gains for each RRF, a percentage was computed on the basis of that RRF's proportion of the total flow on each day (Kuhn and others, 2007). This computational technique proved to be problematic when the quantity of RRF released by an entity varied substantially from day to day. Changes were made to the transit-loss accounting program during 2009 to improve individual RRF accounting. To more reliably track and store streambank-storage losses and gains for each RRF entity, the input configuration of RRFs became fixed, and changes in the name and input order of the RRFs could not be made once entered. This modification allowed transit gains and losses to be stored for the entire recovery period for each RRF in every subreach.

The transit-loss accounting program was modified in 2010 to track exchanges of Fryingpan–Arkansas Project water in the Fountain Creek watershed. The Fryingpan–Arkansas Project Act of 1962 (Public Law 87-590) authorized construction of diversion, conveyance, and storage facilities (which

as of 2018 are in place) to divert, transport, and store water from Colorado River tributaries on the western slope of the Continental Divide for use in water-short areas on the eastern slope in the Arkansas River watershed in southeastern Colorado (Southeastern Colorado Water Conservancy District, 2018). The Project is managed by the Bureau of Reclamation and Southeastern Colorado Water Conservancy District and currently diverts an average of about 57,400 acre-feet of water each year from the Fryingpan River watershed (Southeastern Colorado Water Conservancy District, 2018; Roaring Fork Conservancy, 2018). The City of Colorado Springs owns a share of the Project water and is able to exchange RRF flows discharged at its facilities for Project water.

Brian Sutton, Colorado Division of Water Resources, explained the use of Southeastern Colorado Water Conservancy District (SECWCD) Project water in this way. "All of the surface rights on Fountain Creek have the option to purchase Project water from SECWCD. This water is like an insurance policy for dry times when the ditches may be called out of priority. When a ditch is called out and they have SECWCD Project water available to them, they can divert CSU's RRF water at their headgate. When a ditch does this, there are no transit losses associated with the water that was diverted. This results in a net gain to CSU's RRF. SECWCD transfers (exchanges) an equal amount from the SECWCD account into the Colorado Springs' account in Pueblo Reservoir. A "balance account" was established to keep track of this net gain water (NGW) for selected diversion structures that provide such irrigation water. The net gain is split between the balance account (75%) and Colorado Springs' account (25%) until the overall content of the balance account reaches 1,000 Acre-feet," (Brian Sutton, Colorado Division of Water Resources, written commun., November 18, 2009).

Modifications made to the transit-loss accounting program during 2010 enabled computation of the net gain, or transit-loss savings, realized by diverting CSU's RRFs upstream from the mouth of Fountain Creek. The original 2010 computation of NGW used the upstream diversion release value. In 2011, the part of the transit-loss accounting program that calculates the NGW was revised to use the actual diversion value at the headgate.

In 2012, the transit-loss accounting program was modified to provide increased resolution for very small RRFs released into Monument Creek. Linear interpolation was also used to derive additional values for initial streambank storage loss in the lookup tables for RRFs between 0.0 and 5.0 ft^3/s. Between 0.0 and 1.0 ft^3/s, the interpolation was in 0.1 ft^3/s increments; between 1.0 and 4.0 ft^3/s, the interpolation was in 0.2 ft^3/s increments; and between 4.0 and 5.0 ft^3/s, the interpolation was in 0.5 ft^3/s increments. The interpolated values of initial streambank storage loss were incorporated into an auxiliary set of lookup tables for use only along Monument Creek when RRFs are less than or equal to 4.5 ft^3/s. The transit-loss accounting program was modified to (1) read and store the data in the auxiliary table internally (just as is done with the original table) and (2) use the auxiliary lookup tables to estimate streambank losses for RRFs less than or equal to 4.5 ft^3/s and the original lookup tables to estimate streambank losses for RRFs greater than 4.5 ft^3/s.

Between January and April 2012, an auxiliary Fortran program was written, and several modifications were made to existing transit-loss accounting program files to enable input of new RRFs and regrouping of existing RRFs by entity name. The auxiliary Fortran program was written to (1) input a new set of RRF entity names as specified by the DWR District 10 Water Commissioner who operates the accounting program, (2) incorporate the new set of RRF names into the daily data input file, (3) incorporate the new set of RRF names into the "Recovery" file while maintaining the streambank storage loss and gain data for all existing return flows in the file, and (4) incorporate the new set of RRF names into the file that maintains monthly and annual flow release volumes for each individual RRF entity while maintaining these data for all existing RRFs in the file. To enable addition of new RRF entities into the blank data fields in the daily input file, the transit-loss accounting program also was modified. Changes were made in those sections of the transit-loss accounting program that read the daily input data file, in sections that store the data in internal arrays, and in the sections that output the transit loss computation results.

Upgrade of the Transit-Loss Accounting Program from FORTRAN77 to Fortran 2003

Fortran, an acronym for FORmula TRANslation, was one of the first high-level programming languages intended for mathematical computations used in sciences and engineering. Since being developed in the 1950s at International Business Machines Corporation, Fortran has endured as a programming language for well over 5 decades. As of October 2018, five standards have been released. These include FORTRAN77, Fortran 90, Fortran 95, Fortran 2003, and Fortran 2008 (Chapman, 2017). Each new standard evolved by adding new programming features while retaining compatibility with prior versions. To retain compatibility with older standards of the language, each successive standard incorporates the preceding standard, with only a few exceptions involving features that have been removed. The original transit-loss accounting program was written using FORTRAN77.

The Fortran 90 standard, published by the Organization for Standardization in 1991 and American National Standards Institute in 1992 (Adams and others, 1992), is the much delayed successor to FORTRAN77. Fortran 90 removed very few features and added many new features to reflect substantial changes in programming practices. Fortran 90 only depreciated obsolete FORTRAN77 features without actually removing them. Because all FORTRAN77 features are implemented by Fortran 90, FORTRAN77 is considered a true

subset of Fortran 90. Fortran 90 provided the foundation upon which all successive versions of Fortran to date (2018) have been developed. Many features declared to be obsolete can still be included without causing compiler error.

Fortran 95 contained only minor changes to Fortran 90 and eliminated several obsolescent FORTRAN77 features retained in the Fortran 90 standard. Fortran 2003, published in 2004 (Intel, 2012), represented the second major Fortran revision. It introduced a number of new features that enhanced Fortran 90 features. The most substantial improvement added by the Fortran 2003 standard was support of object-oriented programming. Fortran 2008, published in 2010 (Intel, 2012), was a minor upgrade that largely incorporated clarifications and corrections to Fortran 2003. Most importantly, all FORTRAN77 features are implemented by later Fortran standards, with most modern Fortran compilers able to support all existing standards.

For the case of the transit-loss accounting program, the Fortran 2003 standard is the most recent standard that offers the full range of functionality required to successfully execute it. However, only a small subset of Fortran 2003 constructs were incorporated into the upgraded program because many of the newer Fortran constructs had little or no application to the original transit-loss accounting program or would have required extensive re-organization and rewriting of the program. Fortran 2003 features used to upgrade the transit-loss accounting program include lower-case statements; free-form input; an increase in the length of variable names from 6 to 63 characters; inline comments; whole, partial, and masked array operations; modules that allow grouping of related variables and procedures, and better control of variable scope; improved subroutine argument-passing mechanisms; dynamic memory allocation; named ***if*** statements; named structured looping constructs; and construct for multi-way selection.

All 6,038 lines of the original FORTRAN77 transit-loss accounting program were upgraded to the Fortran 2003 standard. Revisions to the original program included changes that improve readability and facilitate future modifications, better protect variables and fixed values, customize memory reserved for each daily run to the exact size of the problem at hand through array declaration and processing, and improve handling of input errors. Upgrades to the program were completed by stepping through each line of the program and altering the format, syntax, and updating to Fortran 2003 programming features and constructs. During the first pass through the original Fortran program, the comment lines and continuation statements were reformatted. During the second pass, all program text was converted to lower case. During subsequent passes, the fixed formatting required by the FORTRAN77 standard was manually changed to free-form input, variables names were lengthened to be more descriptive, inline comments were added, arrays were made allocatable, and whole array operations were added. A new module called ***CommonVariables*** was added to the program and used to declare and initialize common block variables that were then selectively referenced. This was done to allow grouping of related variables and procedures and better control variable scope. The ***do*** loops and ***if*** statements were named; new constructs were added or their syntax was upgraded. Optional subroutine arguments were defined, and fixed-value variables were converted to parameters. The upgraded Fortran 2003 program described in this report is available at https://code.usgs.gov/water/Transit-Loss.

Changes to Improve Readability

Improved readability of the transit-loss accounting program after migration from FORTRAN77 to Fortran 2003 is considered to be the single most important upgrade to the transit-loss accounting program. Despite being heavily documented with comments, parts of the original Fortran program are difficult to follow. When a program is difficult to read, it is difficult to understand and can cause programmers to introduce logic errors to the program. Given that the transit-loss accounting program has been in use for nearly three decades and will likely be used, in some form, for several more decades, any effort to make the code more readable will ultimately make it far easier to maintain. The longer the transit-loss accounting program remains in use, the more likely program revisions will be required to improve model performance, add new functionality, or adapt it to changes in operating systems and computer hardware. Program readability is particularly important if future changes in Colorado's water policy mandate substantial revisions to the transit-loss methodology or as programmers unfamiliar with the program become involved in updating and maintaining the code.

Introduction of modular programming into the accounting program by Kuhn and others (1998) greatly enhanced readability by grouping related, repetitive operations into subroutines. However, subroutines are merely "containers" that help organize similar code. When original code was cut and pasted into subroutines in the 1990s, the revised original program merely became a collection of subroutines, each containing organized clusters of code. Although this made the Fortran program as a whole inherently more readable, code in several of the longer subroutines was often difficult to follow. Despite these challenges, several upgrades that exploit new Fortran 2003 constructs to improve readability were implemented.

Comments and Continuation Lines

The first substantial modification made to the transit-loss accounting program involved changing the FORTRAN77 line continuation format to the Fortran 2003 line continuation format. The FIXCON program (Burkhardt, 2017) was used to upgrade line continuations. In FORTRAN77, a character entered in column six indicates that the current line is a continuation of the previous line. Fortran 2003 extends one line to the next by including an ampersand as the final character, with at least one blank space between the final character in the line and the ampersand.

The FIXCON program was also used to replace the character "C" in column one with a "!". FORTRAN77 programs use a "C" in column one to designate a comment statement, whereas Fortran 2003 programs use an "!" to designate a comment statement. Because comments in the FORTRAN77 transit-loss accounting program were critical to understanding the program, all original comments were preserved and incorporated into the upgraded program. Additional comments were inserted into complex parts of the program to describe program operations in greater detail.

Use of Free Format

One of the most obvious differences between FORTRAN77 and Fortran 2003 is the change from fixed-format code to free-format code. Free-format code provides the programmer with greater latitude over where to type commands than previous iterations. With free-formatted Fortran 2003 code, the programmer is free to type mixed-case comments and commands in columns 1 to 132, can place labels in any columns as long as they precede the statement, can start statements in any column, and can use multiple statements on each line as long as they are separated by semicolons. Although these Fortran enhancements may seem cosmetic, they promote improved maintenance of the transit-loss accounting program because they dramatically improve readability and, as a consequence, help facilitate a better understanding of the program.

Line Indentation

Although indentation could be used in the original FORTRAN77 transit-loss accounting program to enhance its readability, the limitation of 80-column program lines restricts the number of spaces that can be used to delineate blocks of code. Indents of only one or two spaces made it difficult to distinguish between blocks of related code or to highlight lines that are a continuation of a statement from a previous line. More pronounced indentation, using increments of five spaces, was used to emphasize the relation between various parts of the transit-loss accounting program during program upgrade. Close attention was paid to indenting lines of code so that multiple levels of nesting in ***if*** statements and ***do*** loops could be more easily distinguished from each other. Each block of code within an ***if*** statement and ***do*** loop was indented five spaces. Likewise, continuation lines were indented five spaces to highlight the fact that they are not stand-alone statements. Although addition of indented lines in no way altered the speed or accuracy of the program, it greatly improved its readability.

Replacement of Relational Operators

In the interest of making the transit-loss accounting program more readable, FORTRAN77 relational operators *.lt., .le., .eq., .ne., .gt., and .ge.* were replaced with the Fortran 2003 relational operators <, <=, ==, /=, >, and >=, respectively. Although these changes did not improve computational performance, they did make the code more readable and, as a result, easier to maintain.

Named IF and DO Constructs

A number of Fortran 2003 constructs, including ***if*** statements and ***do*** loops, were used throughout the original FORTRAN77 transit-loss accounting program. The ***if*** statements are conditional statements that evaluate a logical expression and transfer program control to other statements, depending on the outcome of the evaluated expression. They are critical to executing complex sequences of commands related to the estimation of return flows under a range of physical conditions. The ***do*** loops cycle through elements in arrays to perform the same mathematical operation on each array element. Arrays that store native flows, releases of imported waters, streambank storage losses, and a variety of other variables required to assess stream transit losses are central to producing accurate estimates of net reusable flows. The ***do*** loops also are used by the accounting program to cycle through arrays associated with streambank loss look-up tables, searching for the set of native and reusable flows that match those estimated for the current subreach.

Although ***if*** and ***do*** constructs are not obsolete, they can interfere with code readability, particularly if there are multiple levels of nesting that can obscure where inner ***if*** statements and ***do*** loops begin and end. Fortran 2003 ***if*** statements and ***do*** loops can begin with a unique descriptive name followed by a colon and end with the same name after the closing ***end if*** or ***end do*** statement. For ***if-else*** constructs, the name was also placed after the ***else*** statement to help identify the ***else*** as belonging to the same block of code as the ***if*** and ***end if*** statements. By assigning names to ***if*** statements and ***do*** loops, the programmer can easily see where long or complex nested ***if*** statements and ***do*** loops begin and end.

DO-EXIT and DO-CYCLE Constructs

Fortran 2003 ***do-exit*** and ***do-cycle*** constructs transfer control to the statement after the end of a ***do*** loop or to the end of a ***do*** loop, respectively. For example, a ***do*** loop reads through all rows of the streambank storage lookup table until it finds the row corresponding to the estimate of the RRF rate in the current subreach. To prevent the program from continuing to read table rows after it found the matching segment, an ***exit*** command was entered to replace the ***goto*** statement. Until a matching row is found, the program will continue looping through all table rows.

The CASE Construct

The ***case*** construct was not available under the FORTRAN77 standard. This construct was used to update

some of the block ***if*** constructs within the transit-loss accounting program to improve readability. Unlike the ***if*** construct, which allows only one logical test to be performed per statement, the ***case*** construct can provide multi-alternative selection. Program variable ***RunOption*** values of 1, 2, 3, and 4, input by the user, control which transit-loss accounting program operation the model will perform. A value of 1 was used to compute transit losses for the current day; a value of 2 was specified when the user wants to recompute transit losses for the previous day. A value of 3 was input to compute transit losses for the day prior to the previous day, and a value of 4 was used to view or update diversion ditch account data. In the original program, a block of ***if if-else if*** statements was used to determine the sequence of statements executed, based on the input run option. This block was replaced with the newer ***case*** construct to allow the program to branch to different sets of commands, depending on the run option specified. Replacing the block ***if*** structure with a ***case*** block did not enhance computational efficiency of the program.

Long Variable Names

Under the FORTRAN77 standard, variable names were constrained to be 1 to 6 characters in length, could not include special characters, and had to begin with a letter. This clearly limited the programmer's ability to assign meaningful variable names within the transit-loss accounting program. Under the Fortran 2003 standard, maximum variable name length was increased to 63 characters. Variable names as long as 61 characters were used to replace the shorter and less descriptive variables used in the original program. All variables and subroutines in the original transit-loss accounting program were renamed using "upper camel case," where the first letter of each word in a compound variable name is capitalized. Use of underscores to delimit words in variable and subroutine names, which are allowed under Fortran 2003 standards, was rejected as an option because it was believed to interfere with program readability.

Changes that Protect Variables and Fixed Values

Some additional Fortran 2003 enhancements that helped to improve readability and simplify the code include removal of implicit typing, introduction of modules to the program, and conversion of fixed-value variables to parameters.

Removal of Implicit Typing

Most programming languages require that all variables be explicitly typed. Fortran is not one of them. It is a weakly typed language that allows the programmer to use implied typing of variables, based on the first letter of the variable name. Any variable not explicitly declared in a type statement is automatically assigned a data type based on this implicit naming convention. Under all Fortran standards published to date, variables that begin with characters I through N or their lower-case equivalents are assumed to represent integer variables, with all other variable names reserved for real-valued or string variables. Although this can be a convenience for programmers, it can lead to serious programming errors if a variable intended to be real valued is accidently named with a character beginning with characters I through N.

Under the FORTRAN77 standard, there were no options available to turn implicit typing off and force the programmer to explicitly declare variable type for all variables. But with later standards, it became possible to control the implicit naming convention using the ***implicit none*** statement. The ***implicit none*** turns off the implicit data typing convention to help guard against errors caused by use of the default declaration. Although ***implicit none*** is not required, it prevents a programmer from using implicit typing rules that can lead to unintentional errors, dramatically reducing the likelihood of errors caused by incorrectly spelled variable names. When ***implicit none*** is included, using any variable without first declaring it explicitly in a type statement will cause an error on compilation. Because the naming convention for ***implicit*** is global, ***implicit none*** only had to be added to the main program to be made visible to all subroutines. When ***implicit none*** was added to the beginning of the main original transit-loss accounting program, it was discovered that four integer variables and one real variable were not explicitly typed. Fortunately, under the implied typing rules, all five variables were correctly typed.

Conversion of Fixed-Value Variables to Parameters

Several variables were set with fixed values or as model constants within the original transit-loss accounting program. These variables were declared as parameters, also referred to as named constants, within the ***CommonVariables*** module at the top of the upgraded Fortran 2003 program. These values include minimum RRF for using the bank-storage lookup table, minimum RRFs in Monument Creek and Fountain Creek below which streambank storage is assumed equal to 0, unit conversion factors, slopes and intercepts associated with regression of flow versus stream width and required for estimation of evaporation losses, and percent convergence closure tolerance. In the original FORTRAN77 program, modification of these constant values required the programmer to perform a global search and replace, a potential source of computational error if the programmer failed to update all instances of the fixed-value variable. In an effort to streamline the code, all constants were converted to parameters and declared at the top of the program within the module ***CommonVariables***, a module created explicitly to declare and initialize variables. Because all variables declared in a module have global scope, these parameters are accessible from any subroutine. In addition to making the code more readable and easier to customize, an important benefit of declaring fixed-value variables as parameters is that a standard Fortran compiler will flag any programming changes to the value of the parameter made within the code.

Incorporation of External Common Blocks

In the original Fortran program, variables used to define properties of the physical model were declared in named common blocks stored in external files. These variables were selectively referenced through use of the ***include*** command. The common blocks were selectively added to the main program and subroutines while limiting their exposure to unanticipated changes. In the current upgrade, variables are instead declared and defined in the ***CommonVariables*** module at compile time and made potentially available to all program units by adding ***use CommonVariables*** to the top of every subroutine. Global variables defined in ***CommonVariables*** were selectively referenced in a list passed to the ***use*** command via the ***only*** option. Use of the ***only*** option limited the scope of variables in subroutines, providing the same protection to all variables that was provided by the original accounting program.

Removal of Global Variables as Subroutine Arguments

Scope plays a critical role in protecting variables from accidental change. If a variable is declared in the main program, it has global scope and can be accessed anywhere in the program. The scope of a variable declared in a module is local unless that module is referenced in another part of the program with the ***use*** command. For the case of subroutines, the scope of a variable is local to the subroutine in which it is declared and not accessible outside that subroutine unless a ***use*** statement referencing the subroutine is added.

For many subroutines in the original program, global variables defined in the main program by common blocks are also passed as arguments into the subroutine. Although this is not in violation of programming rules, passing of global variables as subroutine arguments is redundant. If the variable has global scope, it is already visible to the subroutine and does not need to be passed explicitly into that subroutine as an argument. Passing of global variables also creates unnecessary clutter in subroutine calls. As part of the upgrade, all global variables were removed from calls to subroutines, as well as from the argument list in the subroutine statement.

Use of the INTENT Attribute

By default, any variable can be altered within a subroutine. Unintended changes to variables can be controlled through use of the ***intent*** attribute, when passing a variable to a subroutine, either explicitly as a subroutine call argument or implicitly as a global variable declared in module ***CommonVariables***. Use of ***intent*** (***in***) in a variable declaration allows only the value of the variable to be passed into the subroutine. Any attempt to change its value within the subroutine will trigger a compile-time error. When ***intent*** (***out***) is specified, the value of the variable passed as an argument to the subroutine is ignored. Its value is instead assigned independently of the value that was passed in. The default behavior, which can be explicitly invoked using ***intent*** (***out***), allows the incoming value of the variable to be visible to the subroutine but also allows any computation result to be passed back out of the subroutine. Declaration ***intent*** (***inout***) can be used to receive an incoming variable, modify it within the subroutine, and pass the new value of the variable out of the subroutine. Good programming practice dictates that all variables passed as arguments into subroutines have a specified intent.

Use of Modules to Encapsulate Variables

Fortran 2003 supports modules, which act like containers that bundle similar variables or group similar subroutines, helping to make the code more manageable. Modules also protect variables from being inadvertently changed by sharing them only among subroutines that operate on them. Module ***CommonVariables*** was created to declare and assign all variables previously included in the original program using common block files. This module was then referenced in the main program and subroutines by the ***USE*** command, which makes all module variables visible to all subroutines referencing the module. All variables assigned values in a data block within the ***CommonVariables*** module are initialized at compile time. These variables are largely one-dimensional arrays that store physical properties corresponding to the network of nodes and subreaches and other fixed characteristics of the underlying physical model.

Subroutines, which can be viewed as a type of module, are containers that hold statements performing related operations. To limit the scope, some are declared locally in the subroutine. This ensures that their values are never visible outside the subroutine unless explicitly passed as an argument in a call to another subroutine.

Changes in Array Declaration and Processing

In addition to the six simple data types used to define scalars, the Fortran 2003 standard also supports arrays. Arrays are data structures that store a fixed number of data values of the same type. With the exception of the array containing the bank-storage lookup table values, all arrays used in the accounting program are one-dimensional arrays that contain properties or estimated flows for stream segments and subreaches. Array processing capabilities make it possible to work directly with a whole array or an array section without including the programming steps of ***do*** loops. Intrinsic functions can now act elementally on arrays, and functions can be array valued. Also available are allocatable arrays, assumed shape arrays, and dynamic arrays.

Dynamic Array Allocation

When a scalar variable is declared in a Fortran program, the compiler associates it with a memory location. Likewise, the compiler reserves multiple memory locations for declared array variables. Under the old FORTRAN77 standard, memory could be allocated only during compile time, often forcing programmers to over-dimension arrays in order to ensure that they are sufficiently large to store all array values if more RRF releases or delivery nodes are added during subsequent runs. Fortran 90 and all subsequent standards augment FORTRAN77 with dynamic storage allocation and whole array operations. Under Fortran 90 and later standards, it became possible to dynamically dimension array size during run time using integer variables, allowing program execution to be customized to the exact amount of memory needed to store and operate on array variables. Dynamic array allocation makes it possible to reserve only enough memory to run the transit-loss accounting program for the exact number of RRF releases and diversions specified by the user with no memory wasted.

All arrays declared in the accounting program, except those associated with fixed characteristics of the streamflow-routing model, such as streamgaging stations, stream segments and subreaches, and bank-storage lookup tables, were dynamically dimensioned during runtime. Dynamic dimensioning first involved declaring an array using the ***allocatable*** attribute. The ***allocate*** command then reserves memory during execution by requesting that the operating system remove a block of memory from free store and allocate that block to the executing program. With each allocation, the pool of available memory shrinks.

Whole Array Operations

Unlike FORTRAN77, Fortran 90 and later standards support whole array operations. Whole array operations allow a variety of scalar operations to be applied to entire arrays or to array sections. Where possible, ***do*** loops that step through each array element to assign a value were replaced with whole array assignment statements that simultaneously initialize every element in an array using a statement of the form ***array=initial value***. Such assignment statements were used to initialize each array declared in the ***CommonVariables*** module. Unfortunately, many calls to subroutines in the accounting program pass individual elements of the array, and not the entire array, into the subroutine. This practice precluded use of whole array operations once the arrays were initialized. Full incorporation of whole array operations, as well as array sectioning, would require substantial re-organization of the transit-loss accounting program.

Addition of Input Error Handling

The most frequent source of runtime error during transit-loss accounting program execution occurs as a result of reading input from external files that are incorrectly formatted, do not contain the correct number of records, or are in some way corrupt. In the original transit-loss accounting program, input errors triggered by ***open*** and ***read*** statements were trapped and handled using the ***err*** argument. The ***err=<label>*** specifier only transfers control to the Fortran statement label, like a programmed ***goto*** statement. The ***err*** specifier returns no detailed information about the cause of the input error other than what is provided by the programmer.

To automate reporting of input errors and convey more precise information to the user about the cause of an input error, the ***err=<label>*** specifier was removed from all ***open***, ***read***, and ***write*** statements and was replaced by the ***iostat=<iostatus>*** specifier. This ***iostat*** specifier provides much more detailed information about errors triggered by runtime mistakes in opening, reading, and writing to files. During compilation, 114 possible input/output error codes and their corresponding error message strings are assigned. On execution of an ***open***, ***read, or write*** statement, an integer value of ***iostatus*** is assigned using the ***iostat*** specifier. If the input/output statement executes without error, an error code of 0 is assigned to ***iostatus***. If an input/output error is triggered during runtime, an error code used to report the exact nature of the error is instead assigned (http://msg.ucsf.edu/local/programs/IBM_Compilers/Fortran/html/pgs/lr76.htm, accessed August 30, 2017), and the specific reason for the input error is reported by issuing a call to a new error subroutine. Code within that subroutine loops through all possible integer values of ***iostatus*** until a match is found in the one-dimensional array storing all 114 error codes and their string descriptors. The corresponding string descriptor for the error code is then output to text and .csv output files. With inclusion of the ***iostat=<iostatus>*** specifier in all ***open***, ***read***, and ***write*** statements, it became possible to report all runtime input/output errors to the user and trap these errors for further action. Although negative ***iostatus*** values are included in the array of input errors, errors that trigger values of ***iostatus***< 0 automatically redirect the program to the next program statement. Only values of ***iostatus*** greater than 0 are reported to the user because they indicate more serious runtime errors. The program automatically terminates on any error that reaches this subroutine.

Verification of Upgraded Fortran Program

Detection of logic errors that might have been inadvertently introduced to the transit-loss accounting program was

an important part of the upgrade effort. Such errors differ from compile time syntax errors, which prevent compilation of the program to an executable file. Logic errors also differ from runtime errors that cause the run to abort without producing any output. Unlike compile time and runtime errors, which are immediately evident when an effort is made to compile or run the program, logic errors are far more difficult to detect because they generally do not prevent the program from compiling and running to completion. Logic errors can arise in any number of ways, including accidental change in the order of executed statements, omission of a call to a subroutine, or incorrect entry of an assignment statement. To prevent the occurrence of logic errors, the accounting program was modified incrementally, and verification was performed at regular intervals using the most current version of the upgraded program. Upgrades resumed only after any logic errors introduced to the current version of the Fortran 2003 program were fully resolved. Note that verification will not account for undiscovered bugs in the upgraded program that may occur under flow conditions that rarely occur because those bugs also would have been present in the original program.

Initial verification involved determining whether archived historical daily output generated by the original FORTRAN77 program from January 5, 2015, to October 31, 2018, were reproduced by the upgraded program. Even though the archived output from January 5, 2015, to October 31, 2018, likely does not represent the full range of hydrologic conditions that have prevailed in the study area over the nearly 3 decades the program has been in use, it was considered sufficient to assess whether any logic errors were introduced to the Fortran program during program upgrades. Such errors in logic would generally manifest as substantial differences between estimated net RRFs predicted by the original program and its upgraded equivalent.

Generation of daily output that could be compared to historical output was somewhat complicated by the fact that interim files containing information about antecedent streambank storage losses and recovery were not saved for any historical runs. Without access to these supporting files, the underlying physically based model executed by the upgraded Fortran 2003 program, in effect, had no memory of prior conditions nor any means to compare prior streambank storage and recovery of daily flows. To overcome the problem of missing historical bank storage and recovery files that contain important information relating to antecedent streambank storage conditions, daily runs were made for a period of 104 consecutive days before daily output was compared to historical output. This “spin-up” period was the length of time over which the model retains any memory of previous streambank storage conditions and was determined to be 104 days rather than 60 days, the longest recovery period used in the transit-loss accounting program. This discrepancy can be traced to the fact that Kuhn (1988) and Kuhn and Arnold (2006) defined subreach recovery periods as the amount of time required to recover all but the last 8.4 percent of water previously lost to the streambank. Return of the residual 8.4 percent of streambank storage losses to subreach channels draining the study area added an additional 44 days to the time needed to remove all memory of arbitrary initial conditions used to initialize the daily runs. Therefore, a 104-day spin-up period was required before RRFs calculated by the upgraded program and the original program matched, indicating that the underlying stream-aquifer system is affected by the arbitrary choice of initial conditions for more than 3 months.

The verification process demonstrated that, after the 104-day spin-up period, the daily delivery node and subreach return flows calculated using the upgraded Fortran 2003 transit-loss accounting program matched those output by the original FORTRAN77 program with an accuracy of less than 0.01 ft^3/s and 0.0001 ft^3/s, respectively. The different accuracies used for delivery node and subreach return flows stem from the different numbers of significant digits used to report RRFs in the output files.

Results of the program verification are presented as identity plots of delivery node and subreach RRFs output by the upgraded program in relation to return flows output by the original unmodified program. Such one-to-one plots provide visual confirmation that the upgraded transit-loss accounting program reproduces historical output and that no logic errors were introduced to the Fortran transit-loss accounting program during the upgrade. A plot of delivery node return flows output by the upgraded program in relation to delivery node return flows read from historic output produced on the release days by the original program is shown in figure 2. Similar plots of subreach RRFs for Fountain Creek and Monument Creek are presented in figures 3 and 4. In every case, all points corresponding to estimated return flows fall directly on the identity (1:1) line.

Figures 5 through 7, which show deviations from the identity line for delivery node RRFs and subreach RRFs along Fountain and Monument Creeks output using the upgraded and original programs, confirm that all deviations of predicted RRFs output by the upgraded transit-loss accounting program from flows output by the original program are 0.0 ft^3/s. Deviation of 0.0 ft^3/s indicates that all RRFs are accurately reproduced.

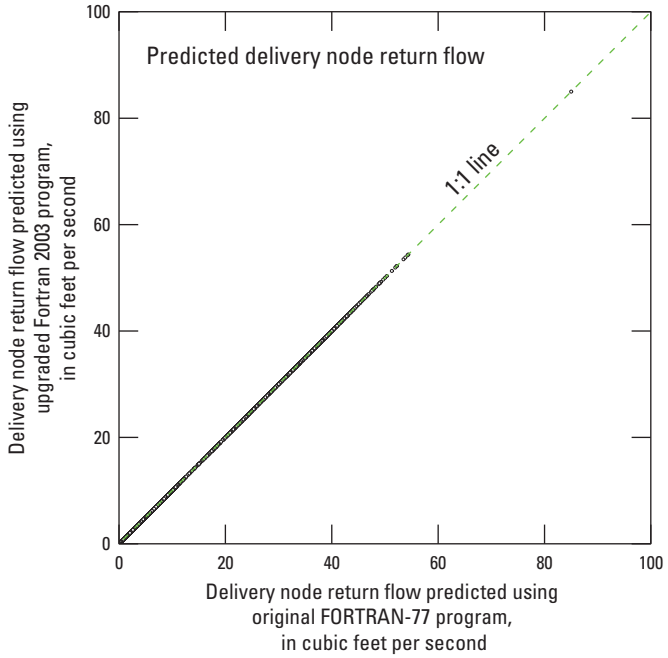

Figure 2. Delivery node return flow predicted using upgraded Fortran 2003 program in relation to delivery node return flow predicted using original FORTRAN77 program.

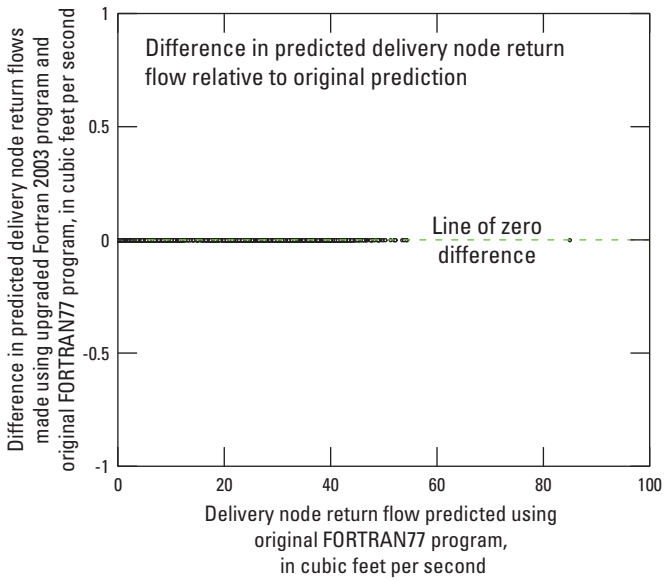

Figure 3. Difference in predicted delivery node return flows made using upgraded Fortran 2003 program and original FORTRAN77 program relative to delivery node return flows predicted using original FORTRAN77 program.

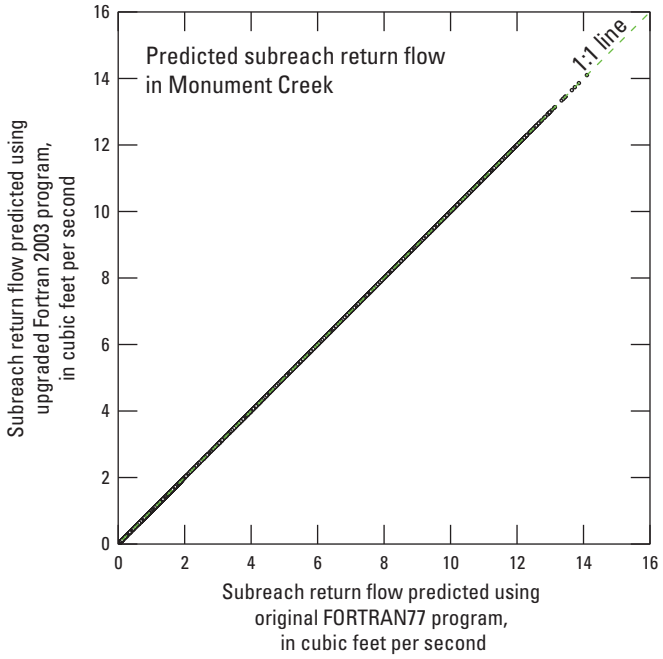

Figure 4. Subreach return flow predicted using upgraded Fortran 2003 program in relation to subreach return flow predicted using original FORTRAN77 program in Monument Creek.

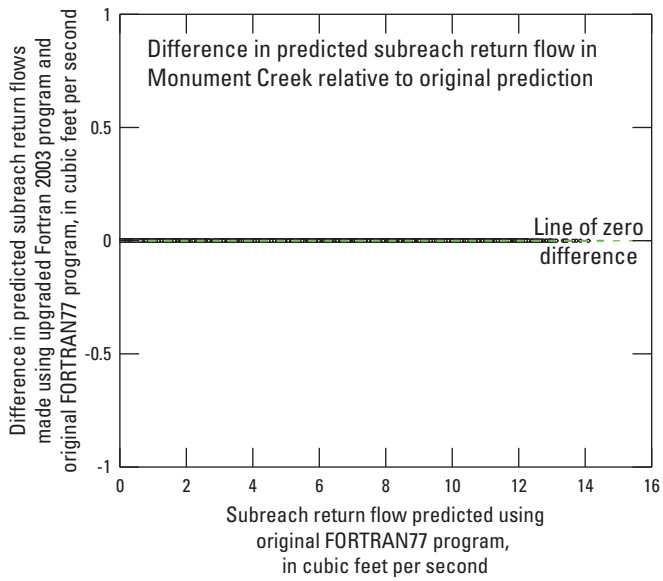

Figure 5. Difference in predicted subreach return flows made using upgraded Fortran 2003 program and original FORTRAN77 program relative to subreach return flows predicted using original FORTRAN77 program in Monument Creek.

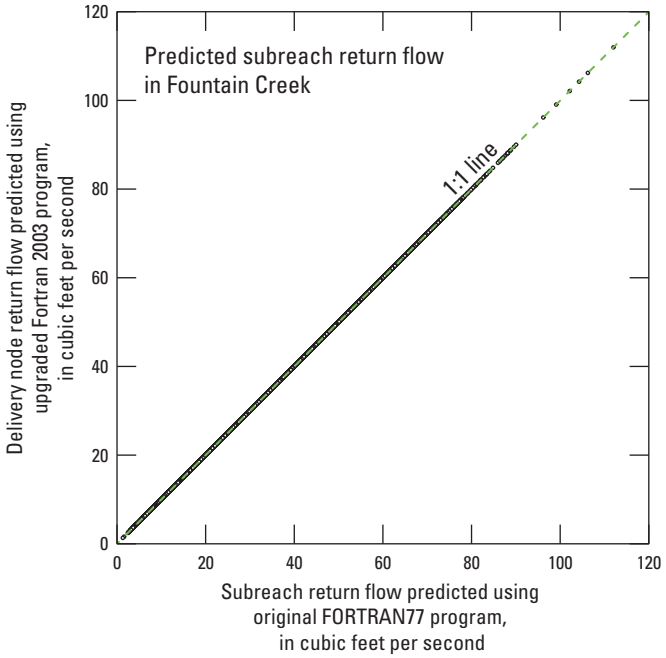

Figure 6. Subreach return flow predicted using upgraded Fortran 2003 program in relation to subreach return flow predicted using original FORTRAN77 program in Fountain Creek.

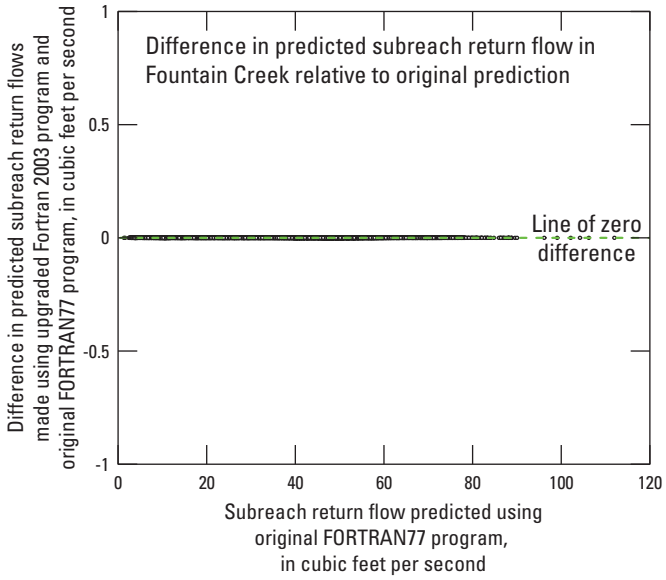

Figure 7. Difference in predicted subreach return flows made using upgraded Fortran 2003 program and original FORTRAN77 program relative to subreach return flows predicted using original FORTRAN77 program in Fountain Creek.

Summary

Under Colorado water law, waters not native to a watershed may be claimed for reuse as long as they can be distinguished from native streamflows. Non-native water imported from outside a watershed can be used and reused to the point of extinction if the amount of imported water can be quantified at its points of intended reuse. A study was completed by the U.S. Geological Survey in 1987, in cooperation with Colorado Springs Utilities, to develop a physically based methodology able to distinguish native (waters naturally occurring in a watershed) from non-native waters (water imported into the watershed from another watershed). The methodology developed as part of this study also accounts for temporary streambank losses to alluvial aquifers that are hydraulically connected to Fountain and Monument Creeks in El Paso and Pueblo Counties, temporary losses to channel storage, and permanent evaporation losses from the stream surface. The methodology was incorporated into a transit-loss accounting computer program.

The transit-loss accounting program, which has been in continuous daily use in some form since April 1989, has provided water-rights administrators with a tool to effectively manage reusable return flow (RRF) and administer water diversion priorities along Fountain Creek on a daily basis. Over time, this transit-loss accounting program has been expanded to include Monument Creek, a tributary of Fountain Creek, and incorporate and track more RRFs from other entities in the watershed. Program updates are needed periodically to make modifications and maintain compatibility with evolving computer software and hardware. To modernize the transit-loss accounting program and ensure compatibility with evolving software and hardware, the U.S. Geological Survey, in cooperation with Pikes Peak Regional Water Authority and the Colorado Water Conservation Board, began an effort in 2016 to upgrade the existing FORTRAN77 transit-loss program so that it complies with more recent Fortran 2003 programming standards.

All 6,038 lines of the original FORTRAN77 transit-loss accounting program were upgraded to the Fortran 2003 standard. Revisions to the original program included changes that improve readability and facilitate future modifications, better protect variables and fixed values, customize memory reserved for each daily run to the exact size of the problem at hand through array declaration and processing, and improve handling of input errors. Upgrades to the program were completed by stepping through each line of the program and altering the format, syntax, and updating to Fortran 2003 programming features and constructs. During the first pass through the original Fortran program, the comment lines and continuation statements were reformatted. During the second pass, all program text was converted to lower case. During subsequent passes, the fixed formatting required by the FORTRAN77 standard was manually changed to free-form input, variables names were lengthened to be more descriptive, inline comments were added, arrays were made allocatable, and whole array operations were added. A new module called ***CommonVariables*** was added to the program and used to declare and initialize common block variables that were then selectively referenced. This was done to allow grouping of related variables and procedures and better control variable scope. The ***do*** loops and ***if*** statements were named, new constructs were added or their syntax was upgraded. Optional subroutine arguments were defined, and fixed-value variables were converted to parameters.

Improved readability of the transit-loss accounting program after migration from FORTRAN77 to Fortran 2003 is considered to be the single most important upgrade to the transit-loss accounting program. Given that the transit-loss accounting program has been in use for nearly three decades and will likely be used, in some form, for several more decades, any effort to make the code more readable will ultimately make it far easier to maintain. Under the FORTRAN77 standard, variable names were constrained to be 1 to 6 characters in length, could not include special characters, and had to begin with a letter. Under the Fortran 2003 standard, maximum variable name length was increased to 63 characters. Variable names as long as 61 characters were used to replace the shorter and less descriptive variables used in the original program. Some additional Fortran 2003 enhancements that helped to improve readability and simplify the code included removal of implicit typing, introduction of modules to the program, and conversion of fixed-value variables to parameters.

Verification of the upgrades focused on reproducing daily estimated RRFs from January 5, 2015, to October 31, 2018. Verification was somewhat complicated by the absence of archived interim support files that provide information about antecedent flow and streambank storage from one daily run to the next.

To overcome the problem of missing historical bank storage and recovery files that contain important information relating to antecedent streambank storage conditions, a 104-day “spin-up” period was required before RRFs calculated by the upgraded program and the original program matched, indicating that the underlying stream-aquifer system is affected by the arbitrary choice of initial conditions for more than 3 months. After this 104-day period, delivery node and subreach return flows estimated using the upgraded transit-loss accounting program and those calculated using the original transit-loss accounting program were equal to one another within 0.01 ft^3/s and 0.0001 ft^3/s, the output reporting accuracy for delivery node and subreach return flows, respectively.

References Cited

Adams, J.C., Brainerd, W.S., Martin, J.T., Smith, B.T., and Wagener, J.L., 1992, The Fortran 90 handbook: New York, N.Y., McGraw-Hill, 823 p.

Burkhardt, J., ed., 2017, FIXCON program, accessed September 1, 2017 at http://people.sc.fsu.edu/~jburkardt/f_src/fixcon/fixcon.html.

Cain, D., and Edelmann, P., 1986, A reconnaissance water-quality appraisal of the Fountain Creek alluvial aquifer between Colorado Springs and Pueblo, Colorado, including trace elements and organic constituents: U.S. Geological Survey toxic waste—Ground-water contamination program: U.S. Geological Survey Water-Resources Investigations Report 86–4085, 45 p. [Also available at https://pubs.usgs.gov/wri/1986/4085/report.pdf.]

Chapman, S.J., 2017, Fortran for scientists and engineers, Fourth edition: New York, N.Y., McGraw-Hill Education, 1056 p.

Farnsworth, R.K., Thompson, E.S., and Peck, E.L., 1982, Evaporation atlas for the contiguous 48 United States: Washington, D.C., National Oceanic and Atmospheric Administration Technical Report NWS 33, 26 p.

Heath, R.C., 1983, Basic ground-water hydrology: U.S. Geological Survey Water-Supply Paper 2220, 84 p. [Also available at https://pubs.usgs.gov/wsp/2220/report.pdf.]

Intel, 2012, Intel Fortran Compiler XE 13.0 User and Reference Guides accessed December 6, 2018, at http://www.cism.ucl.ac.be/Services/Formations/ICS/ics_2013.0.028/composerxe/Documentation/en_US/compiler_f/main_for/index.htm#GUID-AD81102A-2F06-42EE-AA0B-AD8872E43A07.htm

Jenkins, E.D., 1964, Ground water in Fountain and Jimmy Camp Valleys, El Paso County, Colorado, with a section on Computations of drawdowns caused by the pumping of wells in Fountain Valley by R.E. Glover and E.D. Jenkins: U.S. Geological Survey Water-Supply Paper 1583, 66 p. [Also available at https://pubs.usgs.gov/wsp/1583/report.pdf.]

Kuhn, G., 1988, Methods to determine transit losses for return flows of transmountain water in Fountain Creek between Colorado Springs and the Arkansas River, Colorado: U.S. Geological Survey Water-Resources Investigations Report 87–4119, 183 p.

Kuhn, G., and Arnold, L.R., 2006, Application of a stream-aquifer model to Monument Creek for development of a method to estimate transit losses for reusable water, El Paso County, Colorado: U.S. Geological Survey Scientific Investigations Report 2006–5184, 111 p., accessed February 26, 2018, at https://pubs.usgs.gov/sir/2006/5184/pdf/SIR06-5184_508.pdf.

Kuhn, G., Krammes, G.S., and Beal, V.J., 2007, Description and user manual for a Web-based interface to a transit-loss accounting program for Monument and Fountain Creeks, El Paso and Pueblo Counties, Colorado: U.S. Geological Survey Scientific Investigations Report 2007–5028, 36 p., accessed February 26, 2018, at https://pubs.usgs.gov/sir/2007/5028/.

Kuhn, G., Samuels, E.L., Bemis, W.D., and Steger, R.D., 1998, Description of the program changes (1989–97) and a user manual for a transit loss accounting program applied to Fountain Creek between Colorado Springs and the Arkansas River, Colorado: U.S. Geological Survey Open-File Report 97–637, 39 p. [Also available at https://pubs.usgs.gov/of/1997/0637/report.pdf.]

Land, L.F., 1977, Streamflow routing with losses to bank storage or wells: National Technical Information Service, Computer Contribution, PB-271535/AS, 117 p.

Livingston, R.K., 1978, Transit losses and traveltimes of reservoir releases along the Arkansas River from Pueblo Reservoir to John Martin Reservoir, southeastern Colorado: U.S. Geological Survey Water-Resources Investigations 78–75, 30 p. [Also available at https://pubs.er.usgs.gov/publication/wri7875.]

Livingston, R.K., Klein, J.M., and Bingham, D.L., 1976, Water resources of El Paso County, Colorado: Denver, Colo., Colorado Water Conservation Board, Water Resources Circular 32, 85 p.

Radosevich, G.E., Nobe, K.C., Allardice, D., and Kirkwood, C., 1976, Evolution and administration of Colorado water law—1876–1976: Ft. Collins, Colo., Water Resources Publications, 280 p.

Roaring Fork Conservancy, 2018, Fryingpan–Arkansas Project: Basalt, Colo., Roaring Fork Conservancey, accessed April 25, 2018, at http://www.roaringfork.org/your-watershed/watershed-facts/transmountain-diversions/fryingpan-arkansas-project/.

Robinson, T.W., 1958, Phreatophytes: U.S. Geological Survey Water-Supply Paper 1423, 84 p. [Also available at https://pubs.usgs.gov/wsp/1423/report.pdf.]

Southeastern Colorado Water Conservancy District, 2018, Fryingpan–Arkansas Project History: Pueblo, Colo., Southeastern Colorado Water Conservancy District, accessed April 25, 2018, at https://www.secwcd.org/content/fryingpan-arkansas-project-history.

Taylor, O.J., 1975, Artificial-recharge experiments in the alluvial aquifer south of Fountain, El Paso County, Colorado: Colorado Water Conservations Board Water-Resources Circular 31, 28 p.

Waskom, R., and Neibauer, M., 2004, Glossary of water terminology: Colorado State University Cooperative Extension, Water Resource Fact Sheet 4.717, accessed May 16, 2006, at http://www.ext.colostate.edu/pubs/crops/04717.html.

Western Regional Climate Center, 2018, Cooperative climatological data summaries: NOAA cooperative stations—temperature and precipitation, accessed March 28, 2018, at https://wrcc.dri.edu/summary/climsmco.html.

Wilson, W.W., 1965, Pumping tests in Colorado: Colorado Water Conservation Board Ground Water Circular 11, 361 p.

For additional information, contact:
Director, Colorado Water Science Center
U.S. Geological Survey
Denver Federal Center, MS-415
Lakewood, CO 80225

or visit our website at: https://co.water.usgs.gov/

Publishing support provided by the
West Trenton Publishing Service Center

ISSN 2328-0328 (online)
https://doi.org/10.3133/sir20185163