Chapter Three. A Map-Based Surface Water Flow Simulation Model

3.1. INTRODUCTION

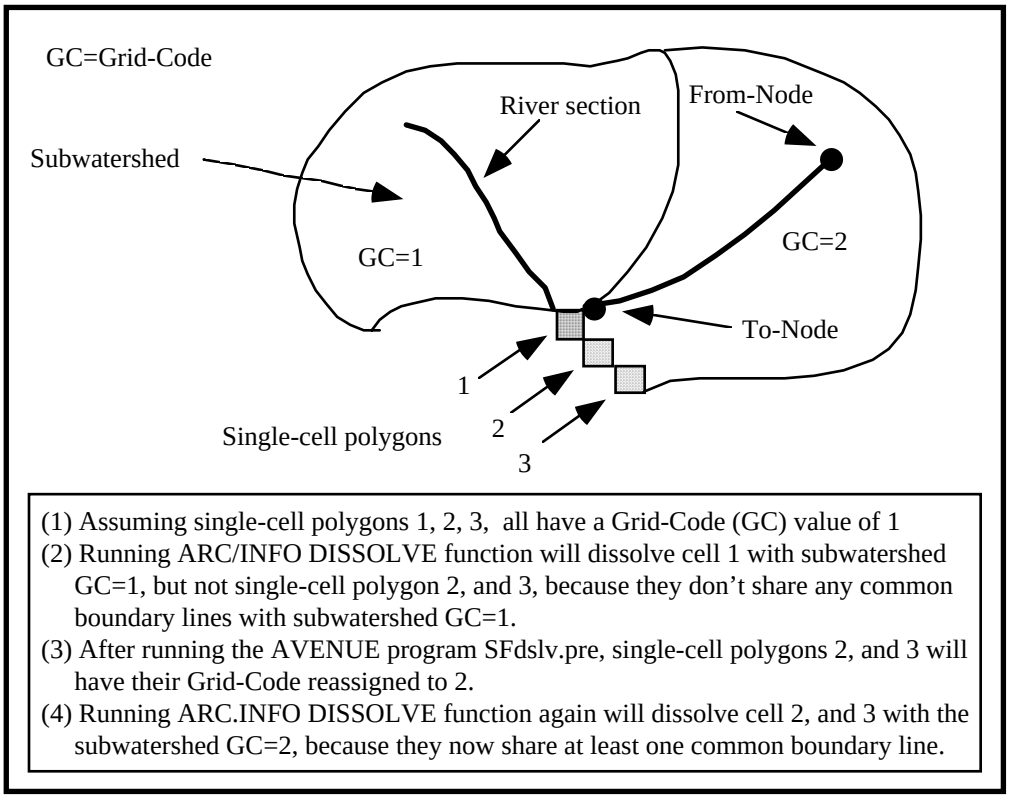
Through the construction of a map-based surface water flow simulation model (SFlowSim) for the Niger River Basin in West Africa, this chapter demonstrates how the three elements (maps, data sets, and programs) of a simulation model are integrated.

The map-based surface water flow simulation model can be used for the water resources assessment and management of a river basin. The model can be applied to any area where a digital elevation model (DEM) is available or to a region whose river basin polygons and river lines are available. Listed below are the tasks that this model can accomplish:

- Simulate river flow time-series based on precipitation defined on watershed polygons or water surplus defined on soil units. After applying the simulation model to a river basin, the flow rates are available at the From-Node and To-Node (termed FFlow and TFlow, respectively, in the model) of each river line. The From-Node and To-Node represent the starting and ending points of a river section line (Figure 3.1). The Post-processor of the simulation model can also interpolate flow rates to any user defined points along the river section.
- Estimate the flow contribution of each subwatershed (termed PFlow in the model).
- Allow reservoir objects to be added to a river section and simulate their effects.

- Allow diversion points to be set on a river section and simulate their effects on downstream river flows.
- Plot a longitudinal flow profiles along a user specified river section.
- Plot the flow time-series (FFlow(t) and TFlow(t)) of a river section or the flow contribution time-series of a subwatershed.
- Allow a user to clip out part of a river basin to create a sub-model so that a more detailed study of the selected subregion can be performed.
- Optimize model parameters to facilitate the calibration of the simulation model.
- Allow a user to modify the modeling conditions directly from the model base maps.
- Integrate with the map-based groundwater simulation model to simulate the flow interaction between surface and subsurface water flows.

Three classes of objects are essential for this map-based surface water flow simulation model. They are (1) a line class created to represent river sections, (2) a polygon class created to represent the subwatersheds associated with the river sections, and (3) a point class created to represent reservoirs or diversion points within any river section. Each river section is an instance (object) of the line class, with its states being stored in a line value attribute table and its behavior described by some flow routing equations. As each record is added to the line attribute table, a new river section object is created. The same thing can be said for the river basin polygon and reservoir point classes.



(1) Assuming single-cell polygons 1, 2, 3, all have a Grid-Code (GC) value of 1
(2) Running ARC/INFO DISSOLVE function will dissolve cell 1 with subwatershed GC=1, but not single-cell polygon 2, and 3, because they don't share any common boundary lines with subwatershed GC=1.
(3) After running the AVENUE program SFdslv.pre, single-cell polygons 2, and 3 will have their Grid-Code reassigned to 2.
(4) Running ARC.INFO DISSOLVE function again will dissolve cell 2, and 3 with the subwatershed GC=2, because they now share at least one common boundary line.

Figure 3.1. Solving the problem of one-line-to-many-polygons

3.2. MODEL CONSTRUCTION PROCEDURE

Based on their functions, the programs in the map-based surface water flow simulation model can be grouped to form three modules: pre-processor, processor and post-processor. The pre-processor is used to create the model objects, construct model base maps, create time-series data tables and process time-series data. The processor is used simulate water flow on the rivers and subwatersheds. The post-processor is used to analyze and display model results. The post-processor also contains utility programs that can be used to modify model maps and modeling conditions and to perform map operation and database

management tasks. Figure 3.2 shows the components of the map-based surface water flow simulation model (SFlowSim) and its construction procedure. The following section discusses the functions of the pre-processor and the construction of model maps.

3.2.1. Preparing Maps for a Map-Based Simulation Model

The maps (river and basin coverages) of a map-based simulation model are constructed by applying the river basin delineation procedure (Maidment, 1994) to a digital elevation model (DEM). As a result of running this delineation procedure, two map coverages are produced. One is a line coverage representing the rivers and the other one is a polygon coverage representing the drainage areas of the rivers. Figure 2.2 shows the rivers and subwatersheds of the Guadalupe River Basin created by delineating a 3" (with a grid cell size of 92.7x92.7 meters) DEM of the region. These river line and subwatershed polygon coverages are imported into an ArcView project and processed by a set of AVENUE programs (pre-processor). The processed river sections form a river network whose members have a one-to-one relation with the subwatershed polygons associated with the river network. The pre-processor programs and simulation model's basic assumptions are discussed below.

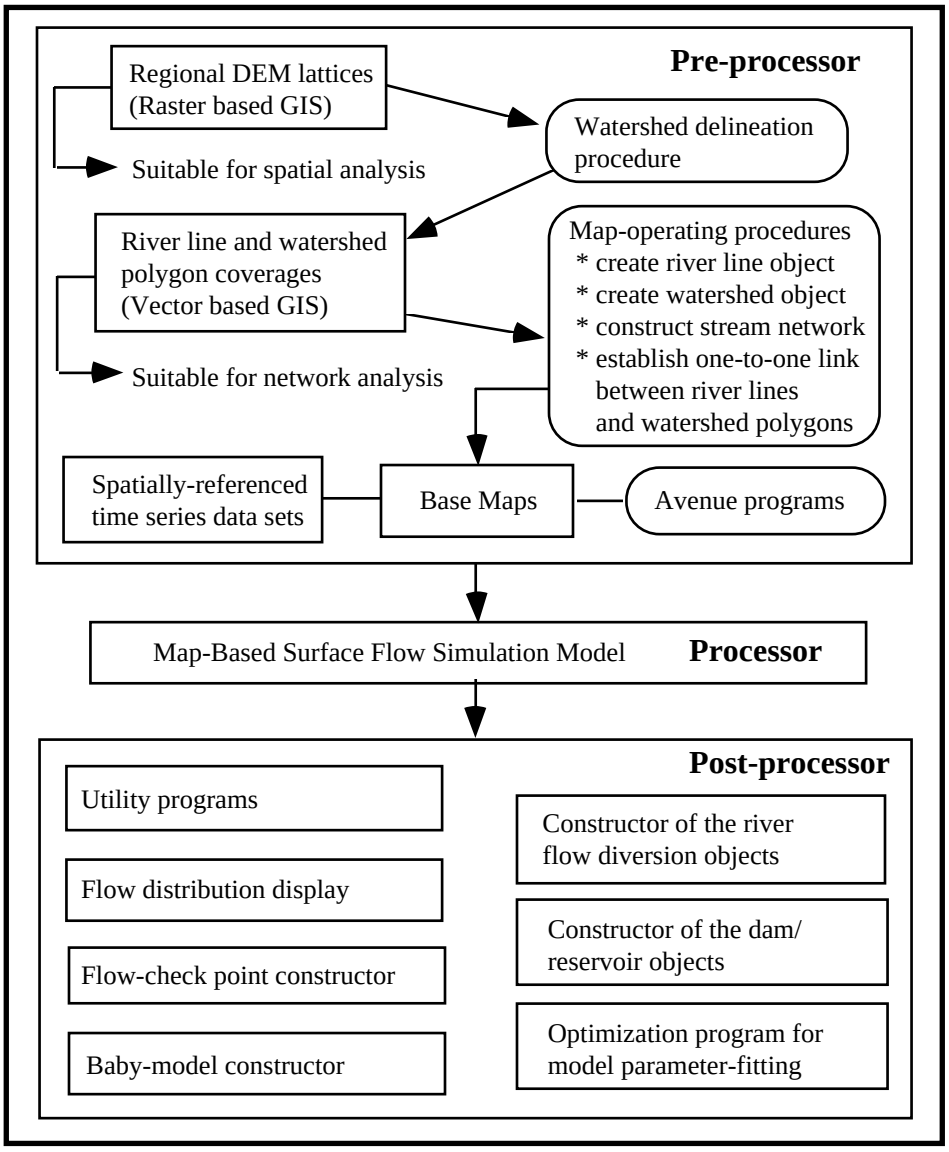


Figure 3.2. Components of the map-based model SFlowSim

3.2.2. Basic Assumptions for a Map-Based Simulation Model

A map-based surface water flow simulation model is constructed based on the following assumptions about the river network and subwatersheds:

(1) Each subwatershed contains one and only one river section,

(2) The simulated quantities such as water flow, chemical mass etc., cannot be transported between subwatershed polygons unless through the river sections that connect them. When the surface water simulation model is integrated with groundwater simulation model in Chapter Five, this condition will be modified to allow water in an integrated model to flow from one subwatershed to another subwatershed through the aquifer underneath them.

(3) For a given river section, the From-Node is always at the upstream end and the To-Node, at the downstream end.

(4) The vertices of a given line[1] are indexed in such a way that a zero index vertex is associate with the From-Node, and for a line of n+1 vertices, the vertex with index n is associated with the To-Node.

Condition (1) is imposed to ensure the one-to-one relationship between the descriptive features of polygon and line coverages. This condition is reasonable because one river subwatershed can contain one and only one river section. Condition (2) is imposed so that network routing is possible. Because river basin boundary lines are formed by the points where elevation potential reaches local maximum, condition (2) is consistent with the definition of a river basin. Condition (2) is therefore reasonable when imposed on water and substances carried by water in a river basin. Condition (3) is imposed to create a river network. Condition (4) is imposed to enable the dynamic segmentation of a line so that any given point on the line can be identified by a fraction representing the ratio between the distance of that point to the From-Node and the total length of the line.

[1] . In ARC/INFO, a line (arc) is represented as a connected set vertices.

3.2.3. Construction of Basic Maps

The vector GIS coverages created by the grid-based watershed delineation procedure (Maidment, 1994) may not meet the assumptions listed above because the raster-to-vector conversion procedure does not always ensure a one-to-one relationship between river lines and subwatershed polygons. The problems to be corrected are usually one-line-to-many-polygons and one-polygon-to-many-lines. The problem of one-line-to-many-polygons is usually caused by the existence of single-grid-cell (single-cell) polygons.

Figure 3.1 shows an example of the problem created by the single-cell polygons. To correct the problem, these single-cell polygons need to be eliminated by dissolving them into the appropriate subwatershed polygons. ARC/INFO provides a DISSOLVE function that will dissolve polygons that (1) have the same value of a given field (e.g. Grid-Code) and (2) share a boundary line. For the single-cell polygons that have the same value of Grid-Code with a subwatershed polygon but do not share any boundary lines with the subwatershed, an AVENUE program is written to reassign to each of these single-cell polygons the Grid-Code of a subwatershed that shares the same boundary line with it. The AVENUE program will first identify and put all single-cell polygons into a list. Then for each member of the list, the program uses LPoly_ and RPoly_ attributes of the polygon's boundary lines to identify one of its adjacent subwatershed polygons, whose Grid-Code value is then assigned to that of the single-cell polygon. The fields LPoly_ and RPoly_ of a line (arc) attribute table hold the machine-assigned IDs of left polygon and right polygon, respectively. After all the single-cell polygons have their Grid-Code reassigned, the DISSOLVE function of the ARC/INFO program is reapplied with Grid-Code as the dissolving

value to eliminate the single-cell polygons. This procedure solves the problem of one-line-to-many-polygons.

The problem of one-polygon-to-many-lines can occur in the situations of (1) split river line segments in a single subwatershed, (2) multiple river line segments within one subwatershed, and (3) incorrect node indexing on the river line coverage. Figures 3.4 and 3.5 illustrate the examples of these situations. One of the tasks of the pre-processor is to eliminate the problem of one-polygon-to-many-lines by running a series of map-processing programs to modify the ARC/INFO coverages created by the watershed delineation procedure. After modification, these two coverages (river line and subwatershed polygon) will meet the four conditions listed in section 3.2.2. and be ready for use by the map-based flow simulation model. The functions of each program in the pre-processor are described below:

1. Creating River Line and Subwatershed Polygon Objects [SFmdfld.pre]

This program adds attributes to the standard ARC/INFO arc attribute table (AAT) and polygon attribute table (PAT) to create river line and watershed polygon objects. The attributes of a watershed polygon object and a river line object are listed in Table 3.1. and Table 3.2. Detailed explanations of these attributes will be given when they are used.

The states of an object are used in this simulation model to describe the physical features of the object and/or to control the processes in the simulation programs. These states can be used in a program as either a regular variable or logical variable or both. When a state of an object is used to describe a physical parameter of the object, it is treated by the simulation program as a regular

variable and the program is interested only in the return values of that state (variable).

Table 3.1. The Attributes of a Subwatershed Polygon Object

	State	**Function (What the attribute represents)**
1	Shape	Pointer pointing to the map location of a polygon object
2	Area	Area of a watershed polygon (m^2)
3	Perimeter	Perimeter of a watershed polygon (m)
4	Cover_	Polygon ID, based on which pointers to time-series vectors (PFlowVt,sprVt, rchVt, headVt, dhVt, dvolVt, etc.) associated with the polygon are constructed.
5	Cover_id	User assigned polygon id
6	Grid_Code	Key field linking a subwatershed polygon with the river line section it contains
7	Pisdone	0 indicates the polygon has NOT been simulated, non-zero, otherwise, and the value equals the number of polygons between this polygon and the outlet
8	PFlow	Local flow contribution, for unsteady state, it gives the average flow rate over the models simulation period (m^3/s).
9	FlowTime	Average time it takes for flow starting from an element (a grid cell) on a subwatershed to reach the outlet point of the subwatershed (s)
10	DiffNum	Diffusion number of PFlow indicating the extend of the PFlow spread-out
11	V	Overland flow velocity (m/s)
12	ThmRslt	For thematic plotting of a selected attribute at a given time step
13	Hasgrd	0 indicates no groundwater flow model exists underneath, 1, otherwise
14	ToGrd	The percentage of flow recharging to the groundwater system
15	MFL	Mean flow length of a subwatershed (m)
16	Msurp	Soil moisture surplus (m^3/s) (subwatershed river flow contribution)
17	ToRes	The fraction of the subwatershed water surplus that goes to subsurface reservoir
18	ResK	Mean residence time of water in a subsurface reservoir [T]

Table 3.2. The Attributes of a River Line Object

	State	Function (What the attribute represents)
1	Shape	Pointer pointing to the map location of an object
2	FNode_	Node ID of the starting point of a river line section
3	TNode_	Node ID of the ending point of a river line section
4	Lpoly_	Left polygon machine-assigned ID (ID of the polygon to the left of the line)
5	Rpoly_	Right polygon machine-assigned ID (ID of the polygon to the right of the line)
6	Length	The length of a river line section (m)
7	Cover_	Machine assigned river line ID
8	Cover_id	User assigned river line ID
9	Grid_code	Key code linking subwatershed polygon with the river line section it contains
10	LIsDone	0 indicates a river line has NOT been simulated, non-zero value indicates otherwise, and the value equals the number of joints between this river line and the basin outlet
11	IsHead	1 indicates a river line is a head section (section with no upstream river lines)
12	IsOutlet	1 indicates a river line is a outlet section (last section on a stream network)
13	FFLOW	The flow rate at the FNode of a river line (m^3/s)
14	TFLOW	The flow rate at the TNode of a river line (m^3/s)
15	DFlow	The water withdraw on a river line (diversion flow rate) (m^3/s)
16	Velocity	Flow velocity on a river line (m/s)
17	LossC	Loss coefficient related to a river line (1/m)
18	Timelag	Flow time to the TNode of a river line along its longest upstream flow path (s)
19	MELE	Mean elevation (definition to be decided) of a river line (m)
20	HasDam	0 indicates there is no dam in the river line, non-zero indicates otherwise, and the value is the dam-id of the first dam on the river line
21	Hasresp	0 indicates no response function is available, non-zero value indicates otherwise, and the value equals the number of the elements in the response function
22	Hasgrd	0 indicates no groundwater flow model exists underneath, 1 indicates otherwise
23	togrd	The percentage of river flow that goes to groundwater recharge

When a state of an object is used by a program for procedure-control purpose, the state is treated by a program as both a logical variable and a regular variable. In this case, the program is interested in both the return values of the state (variable) and the ranges which these values fall into. When a state is used by a program for the purpose of procedure control, a zero-value usually indicates FALSE/NO, a non-zero value usually indicates TRUE/YES. In addition, the non-zero value is also used as a pointer to either a new function, another object, or

values of the physical characteristic of the object. For example, when the state, HasDam of a river line object returns a value zero, it indicates that there is no dam on the river line. When HasDam returns a non-zero value, it indicates there is at least one dam located on the river line and the value is also the identification number (ID) of the first dam on that river line.

Formulating such a design minimizes the number of states of an object so that the number of fields of the database table can be minimize to save the computer memory space and improve program efficiency.

2. Sorting Nodes and Vertices of River Lines [SFsortr.pre]

This program sorts the nodes of river line sections so that the From-Node (FNode) of a river line section is always on the upstream end and To-Node (TNode) is always on the downstream end. The program also sorts the vertices of a river line so that the vertex with zero-index is located at the FNode of the line.

This program first identifies all the river outlet segments and puts the IDs of these river segments into a list. Then, for each member (outlet river line segment) of the list, the program performs the following procedure. Starting with a member on the list, the program traces first in the upstream and then in the downstream direction on the river network with the member as its outlet segment. When moving in the upstream direction, the program travels from arc to arc according to the connectivity established by the FNode and TNode of the arcs (river line sections), corrects the incorrectly indexed nodes, and identifies the river junction sections until a head river section is reached. A head river section is defined as the first river section in a river network, i.e., the river section with no upstream river sections. A junction section is defined as a river section with more than one immediate river sections. The program then starts to move downstream

and stops at each junction section to see if all its upstream river section nodes have been sorted. If not, the program stops moving downstream and starts to move upstream again along another branch of the network that has not yet been checked by the program. The program repeats these upstream and downstream movements until the outlet section is reached.

Figure 3.3 illustrates the program logic and Figure 3.4. shows an example result of applying the SFsortr.pre program to a stream reach network.

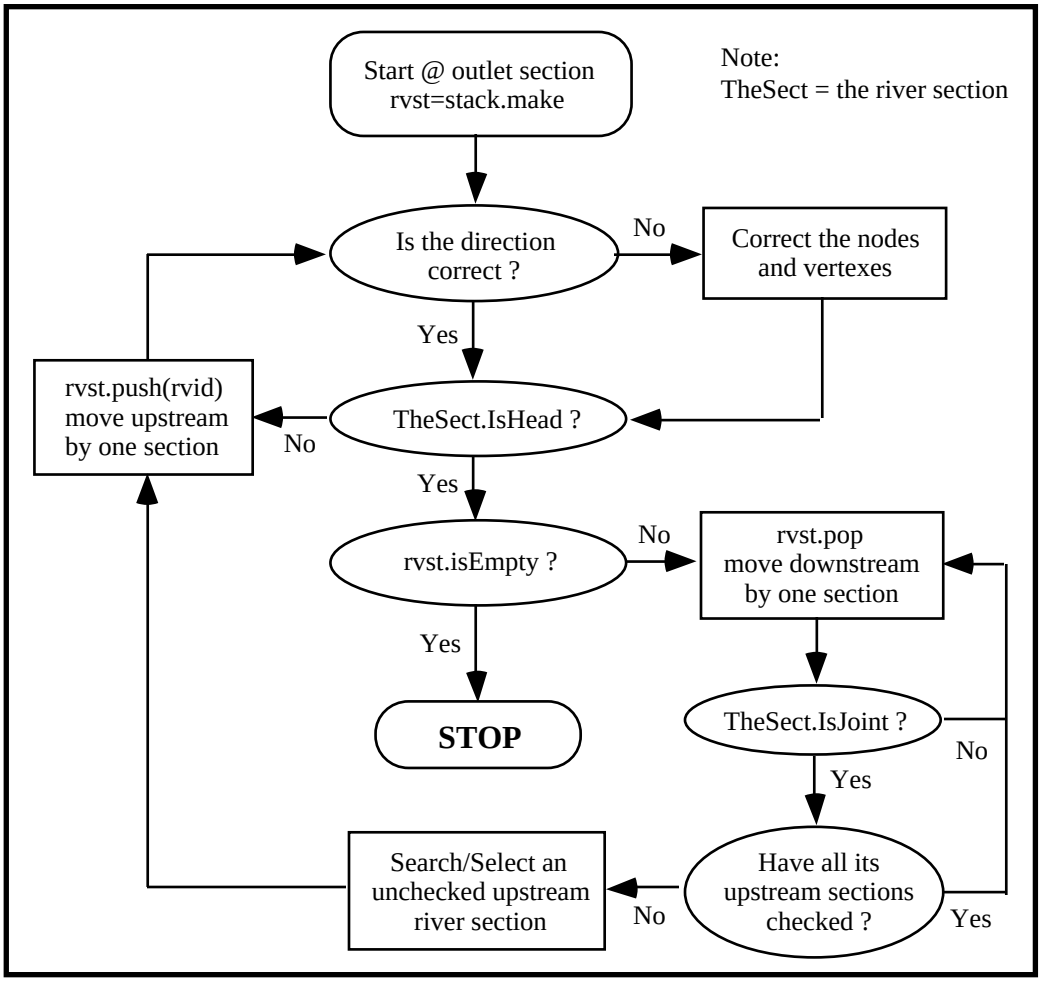


Figure 3.3. Program flow chart for Pre-processor Sfsortr.pre

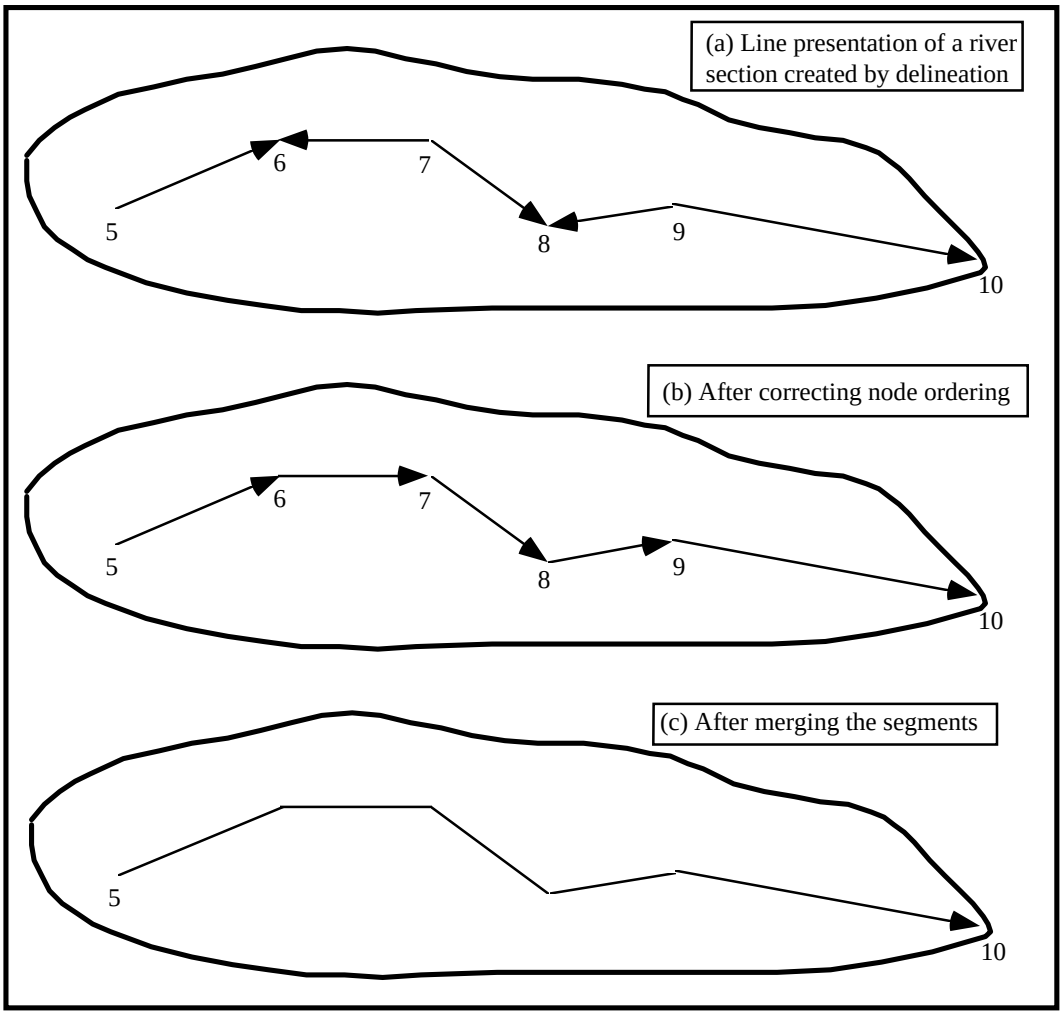


Figure 3.4. Merging multiple river segments into one river section

3. Cleaning the River Line Splits [SFsplit.pre]

The purpose of the river split cleaning program is to remove any river splits that are contained in one subwatershed polygon so that each subwatershed polygon contains only the river segments that can form one single line. A split is defined as three river segments forming an "Y" shape that are contained by a single subwatershed (Figure 3.5). Without correction, the polygon that contains

split river lines will have a one-to-many relation with the river lines. The splits are caused by the conversion procedure used transform a grid coverage to a vector line coverage. When using the GRIDLINE or STREAMLINE functions to convert a delineated grid lattice to a line coverage, the cells (referred to as joint-cells) at the locations where three subwatersheds joint can assume any ID of these three subwatersheds. If the ID-values assigned to the river line segments at the joint cell are inconsistent with those assigned to the subwatersheds, the problem of line split will occur. Although the lengths of these split lines are typically small (1 to 1.414 times the grid cell size), each one of them still forms a record in the spatial database file. Because of their small sizes, it is difficult to detect and correct them manually.

The split-cleaning program works in the following way. For each polygon in the river basin coverage, the program searches through features on the river line coverage to select all the line segments that are contained within the polygon and sends the selected lines to a list. If the number of line segments selected is greater than one, then for each line segment in the list, the program saves its FNode number. This FNode number is then used to select all the line segments in the list that have that FNode as their TNode. If the number of selected lines is greater than one, then an upstream split is found. To correct the problem, for each one of the split segments, its FNode number is used to trace upstream to find its upstream segment. Once an upstream segment is found, its stream ID is used to reset the stream ID of the split river segment. Figure 3.5 illustrates the correction of a split.

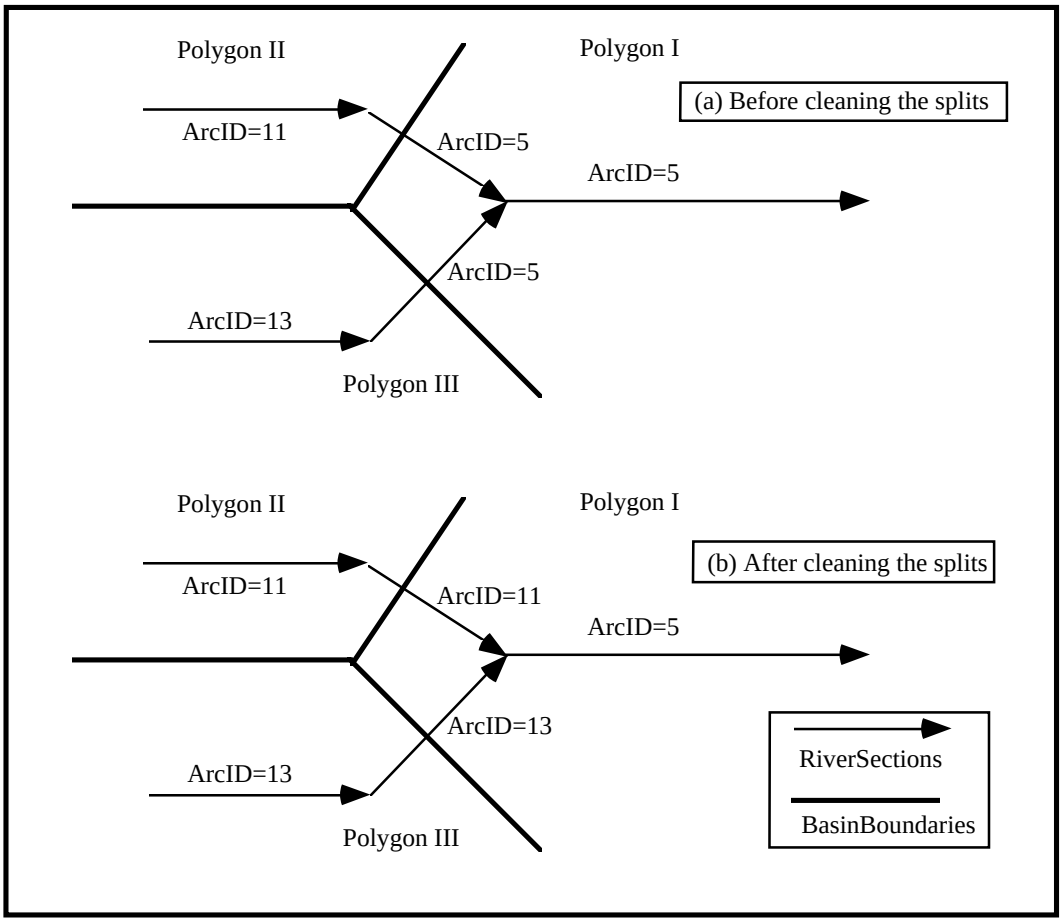


Figure 3.5. Rearranging river sections

3. Merging Multiple River Segments into one River Section [SFmg1ln.pre]

The purpose of the segment merging program is to merge any multiple line segments contained by a single subwatershed polygon into one single line object so that each subwatershed contains only one line section. To do this, the program first creates a new line coverage (NewCov) from the old river line coverage (OldCov). This NewCov is used to hold the river line objects that will be merged. Then, for each polygon in the subwatershed coverage, the program

searches through the river line coverage (OldCov) to select the line segments that are contained by the polygon. If the number of selected records equals one, the program simply copies the river line object to the NewCov database. If the number of lines selected is greater than one, the program will first check through these lines to make sure they are connected to each other in a correct order from upstream to downstream. If they are not, the program will correct the problem based on the TNode and FNode indexes of each line segment. These line segments are then merged and copied to NewCov. To merge these line segments, the nodes and vertices of each line segment are extracted and put into a list based on the order of the line segment. The merged line is then created from this point list.

Upon the completion of the program, SFmg1ln.pre, the NewCov line coverage, and the subwatershed polygon coverage have the following properties: (1) each subwatershed polygon in the basin polygon coverage contains one and only one river line section, (2) each river line section is contained by one and only one subwatershed polygon, (3) the line sections are all connected in the order with FNode on the upstream end and TNode on the downstream end, and (4) all the vertices of a line are correctly indexed with the zero indexed node at the FNode. Figure 3.4 shows an example result of applying SFsortr.pre and SFmg1ln.pre to a river line coverage.

3.3. Database Design for Spatially-Referenced Time-Series Data

Because the simulation model is constructed to simulate surface water flow over both space and time, a database with a data structure that allows efficient storage and retrieval of spatially-referenced time-series data needs to be developed before the simulation model construction can start. This section

discusses the design of such a database and how spatially-referenced time-series data are managed in this map-based surface water flow simulation model.

3.3.1. Two Types of Spatially-Referenced Data

Based on the number of values a physical parameter can have for one location, spatially referenced physical data for a surface and ground water simulation model can be divided into two categories, one value per location (or one-to-one), and many values per location (or one-to-many). The examples of one-to-one data type are: the area of a polygon, the latitude and longitude of a point, or the length of a river section. Time-series data are of the one-to-many data type. Examples include: precipitation records, runoff records, and water level records.

During a modeling process, in order to store and retrieve these two types of data efficiently, different data structures and retrieval methods need to be used. In designing these data structures and data management systems, it is assumed that a location feature, whether it is a polygon region, a line river, or a point runoff station, can be uniquely identified by a location ID (COVER-ID).

3.3.2. Data Structure for Data of One-to-One Type

Data of the one-to-one type can be stored as location attributes attached to a conventional GIS coverage. In this type of database, location ID is used as a key field for data storage and it is also the only information needed to retrieve a data value of a given physical parameter. The data structure of a one-to-one data type is illustrated below in Figure 3.6.

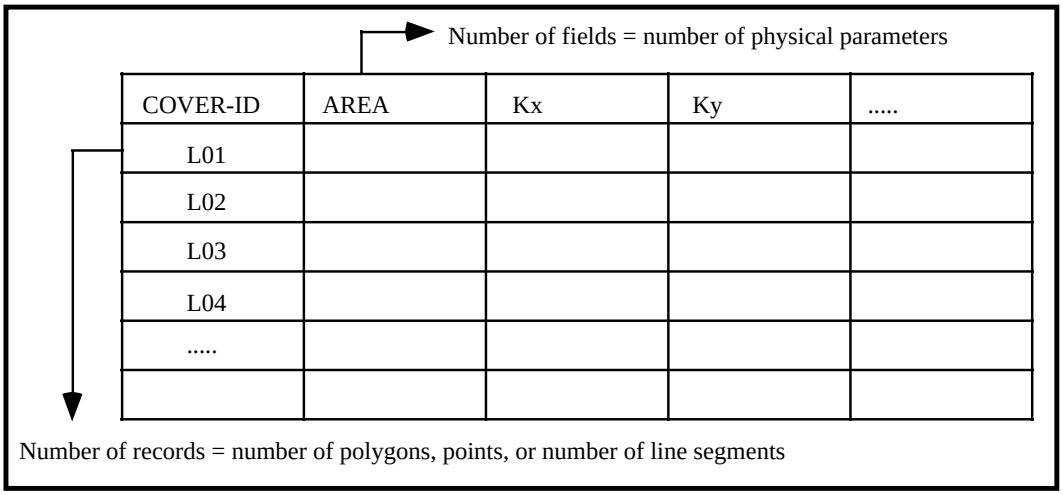


Figure 3.6. Database structure for the one-to-one data type

3.3.3. Data Structure for Data of the One-to-Many Type

Data of the one-to-many type can be stored in separate databases with one database file for each parameter at each location. The file structure of the one-to-many data type is illustrated in Figure 3.7. In this type of databases, a file name should reflect the location ID and the name of the physical parameter that it stores. The number of files for a given parameter will be equal to the number of spatial features where the time-series data are available. However, because the number of spatial features is usually large causing large number of files to be created, this type of data structure is inefficient for data storage and retrieval.

To avoid creating and managing a large number of files, another type data structure is designed and used in this study to manage the spatially-referenced time-series data. According the this design, a single data file is created for each spatially-referenced, time-varying attribute. In the file, the time-series data defined on a map is stored in a rectangular table data structure (i.e. the data structure of a relational-database), in which, the columns (fields) corresponds to

the spatial features on a map and rows (records) corresponds to the time steps. Figure 3.8 illustrates this type of data structure. The data structure is designed in such a way that a one-to-one relationship exists between the fields of a time-series database and the spatial features of the map on which the time-series data are defined. The header (field name) is constructed by concatenating some letters related to the name of the location with a unique location ID. The letters can be the abbreviation of either the name of the attribute or the field name of the location ID in the spatial database table.

The key field for this type of database is TIME for time-series data, and STAGE-ID for multiple-stage data. To retrieve a data value for a given physical parameter, one can first get the Location-ID from an appropriate map coverage, then use the parameter name together with the location ID to open the appropriate file and select the field associated with the location, and finally, use the STAGE-ID or TIME value to locate the correct record for data retrieval.

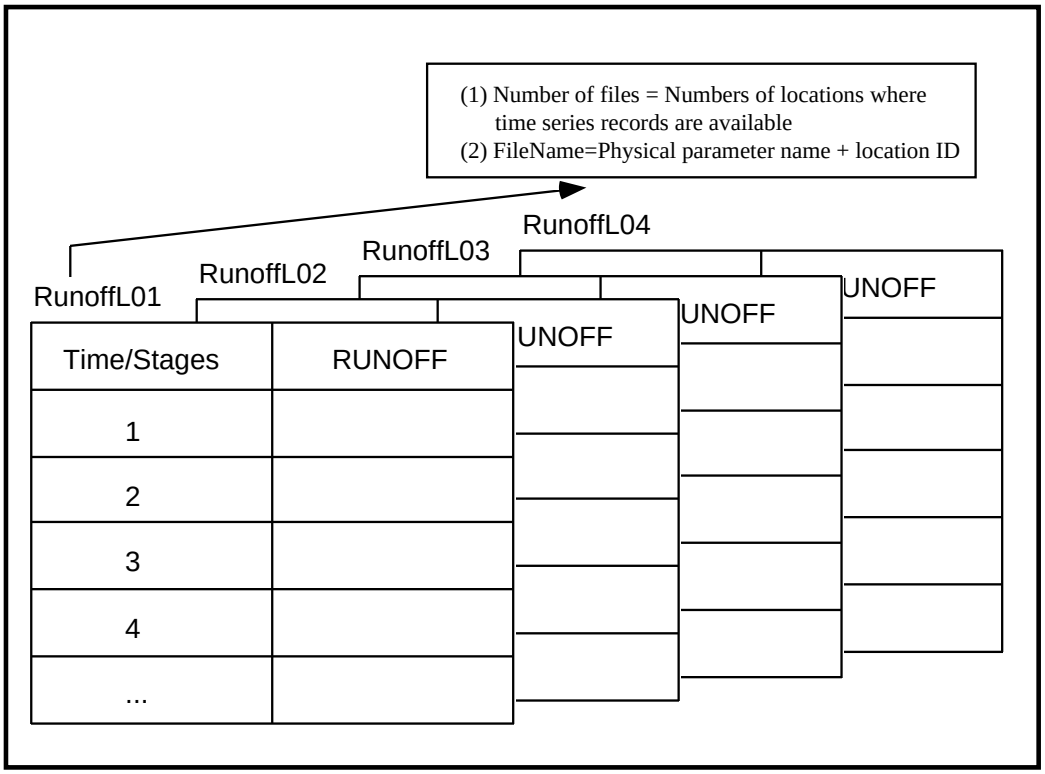


Figure 3.7. The database files for the one-to-many data type

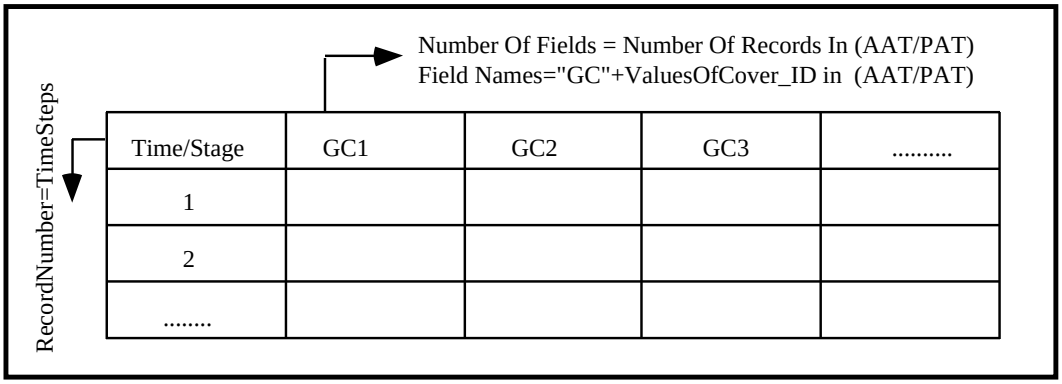


Figure 3.8. The database structure for one-to-many data type

3.3.4. Connections between Databases and the GIS Feature Attribute Tables

In ARC/INFO, feature attribute tables (FTAB) are INFO files associated with each feature type. For a given coverage, a FTAB is created from a template of standard items based on the feature types of the coverage. Figure 3.9 shows the data structures for FTABs associated with each type of coverage.

As stated above, a single database with the template shown in Figure 3.6 can be created to store all the physical data of one-to-one type. The table is then joined through a linking field (usually the COVER-ID) to the FTAB. Once this table merge procedure is completed, the value of a given parameter at any location can be retrieved using the value of the key field (COVER-ID). Because the data items are merged with the value attribute table of a GIS coverage, data retrieval and modification procedures are straightforward.

For the data of one-to-many type, a separate database table with the template shown in Figure 3.8 is created. As this type of database is not physically joined with the FTAB of a map, its data retrieval and modification at a given location need are done through the linkage created by AVENUE programs. To retrieve a time-series record, information about the parameter name, location ID, and the TIME or STAGE-ID are needed. The parameter name and location ID jointly tell the program which database file to open and which field to select, and TIME allows the program to select the correct record number for data retrieval. Figure 3.10 illustrates the data flow path for spatially-referenced time-series data.

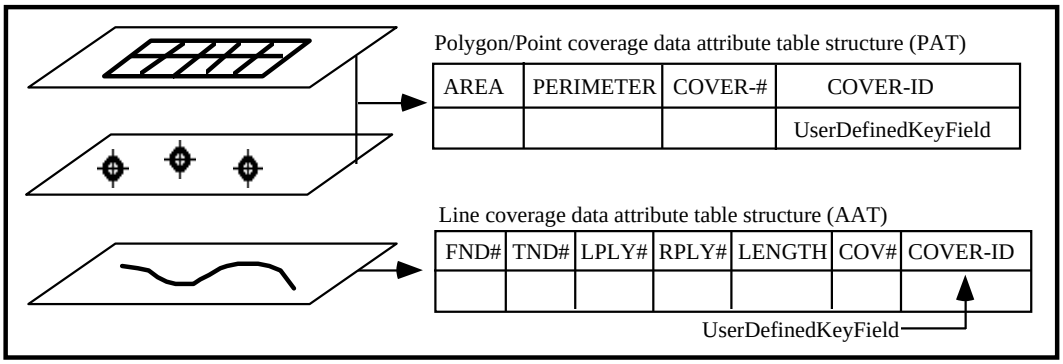


Figure 3.9. Feature attribute tables in ARC/INFO

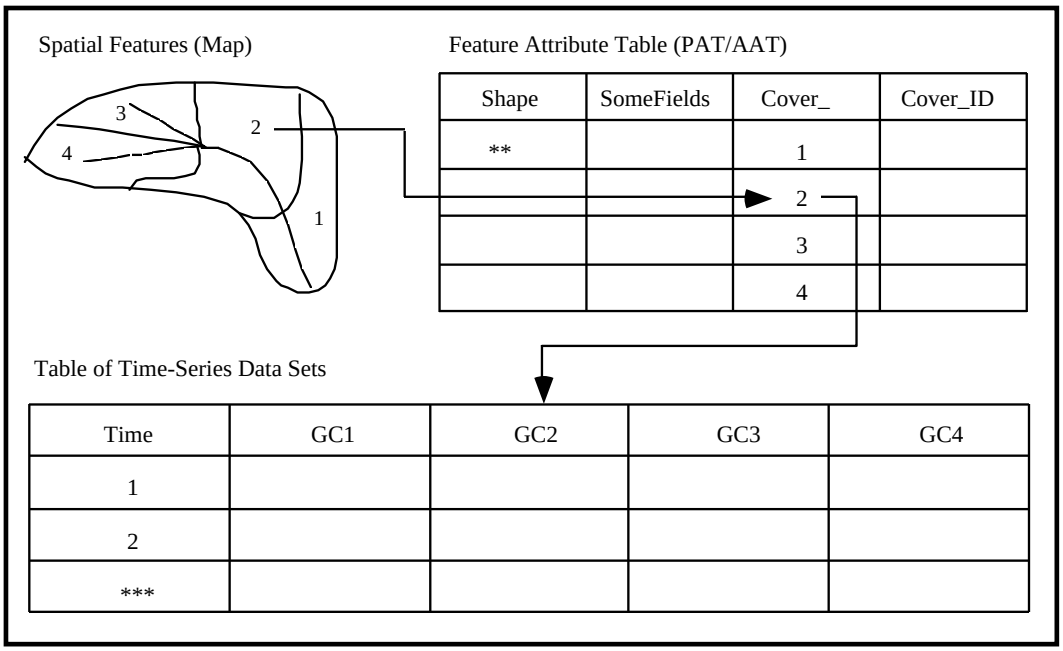


Figure 3.10. Connection between maps and spatially-referenced time-series data

3.4. CONSTRUCTION OF THE SURFACE FLOW SIMULATION PROGRAM

The map-based surface water flow simulation model simulates flow on the river network based on the flow-routing equations introduced in Chapter Two and

the equations to be introduced in the following sections. In the model, water movement in a river network is simulated at three levels: (a) watershed-to-river water movement defined on a subwatershed polygon, which transfers either the precipitation or the soil-moisture surplus into the local stream flow contribution, PFlow(t), (b) water movement on a river section contained in a subwatershed polygon, and (c) water movement between river sections on the river network (inter-section water movements). The algorithm that simulates the water movements between river sections also performs network analysis on the river network to determine an appropriate sequence for simulating each subwatershed. The algorithms that simulate water movements (a) and (b) are problem dependent and are flexible. For this reason, this map-based surface water flow simulation model can easily be modified to simulate other water flow related phenomena, such as non-point source pollution analysis so long as the simulation module for PFlow(t) is properly modified. The following sections show how water movement is simulated in this map-based surface water flow simulation model.

3.4.1. Simulating Water Movement between River Sections

The following facts and concepts are used to construct the object-oriented algorithm that simulates water movement on the river networks.

(1) A line coverage has been constructed using the method described in Section 3.2.3. to represent the river network.

(2) A polygon coverage has been constructed using the method described in Section 3.2.3. to represent subwatershed polygons.

(3) The features in the river line and subwatershed polygon coverages have a one-to-one relationship so that the features of these two coverages are connected through a common key field.

(4) Corresponding to each feature on the line and polygon coverages there is a record in the river line attribute table (RFTAB) and polygon feature attribute table (PFTAB) associated with their respective coverages. Routing through the river network and subwatersheds corresponds to processing though the records in the RFTAB, and PFTAB.

(5) The water movement on the stream network is based on the principle of continuity (e.g. Equations 3.1a and 3.1b) and the connectivity of the network maintained by the From-Node and To-Node of each river line object.

As shown in Tables 3.1 and 3.2 and in Figure 3.11, the PFlow (PFlow(t) for unsteady state) associated with a subwatershed polygon object is used to represent the object's local flow contribution. FFlow and TFlow (FFlow(t) and TFlow(t) for unsteady state) associated with a river line section are used to represent the river flow rate at FNode and TNode of the river line object. Besides applying the continuity equations (Equations 3.1 and 3.2) to the river network to simulate the water movement between the river sections, this module also determines a the order in which subwatersheds will be simulated.

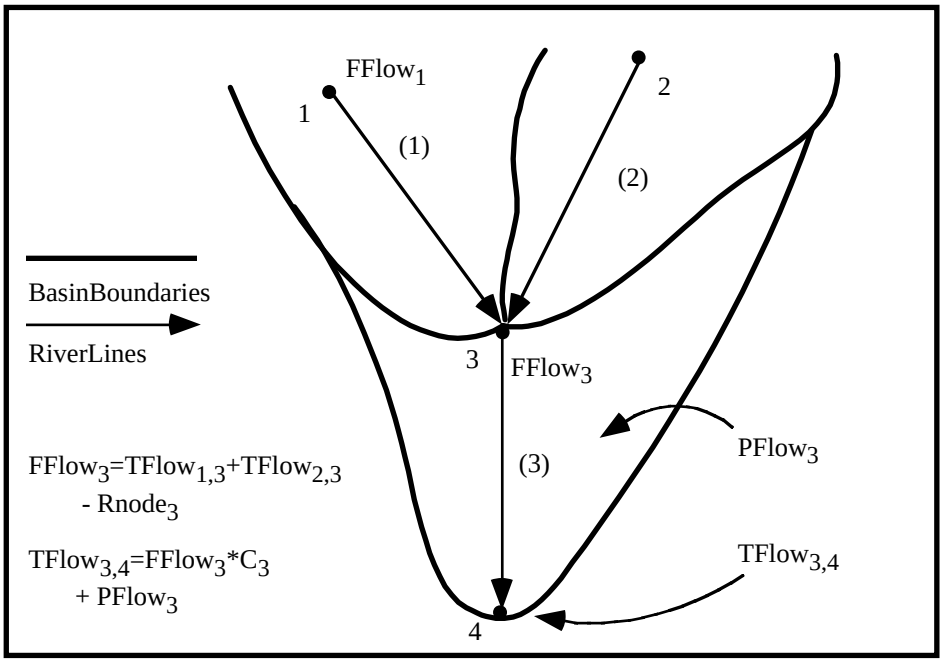


Figure 3.11. River basin flow routing system

The sequential order is constructed using a stack-based algorithm. A stack is an array (collection) that allows the Last-In-First-Out (LIFO) access to its elements. The algorithm is described below (See Figures 3.12a and 3.12b).

When simulating the water movement between the river sections, each stream line and each subwatershed are treated as entities. The effect that a stream section and its subwatershed have on the stream network can be replaced by a value, TFlow of the river section for steady state, or by a vector TFlow(t) for unsteady state. In other words, as soon as TFlow, (or TFlow(t) for unsteady state) of a subwatershed is acquired, the river section as well as the streams upstream of the section can be cut-off from the remainder of the network.

As the algorithms used to simulate the physical processes in terms of PFlow, FFlow, and TFlow depend on the modeling conditions and methodology used, they can be different from one model to another. The flow simulation

algorithms used in this research are presented in sections 3.4.2, and 3.4.3. To illustrate how water movements between river sections are simulated, it is assumed that (1) the PFlow of a subwatershed polygon object is retrievable by sending a request to the polygon object, and (2) FFlow and TFlow can be computed based on the PFlow defined on the subwatershed polygons and the connectivity of the river network. Under steady state, the relationships (Equations 3.1 and 3.2) between FFlow, PFlow, and TFlow can be derived from the principle of mass conservation and the connectivity of the river network (Figure 3.11).

Given a river section *i*, we have:

$$FFlow_i = \sum_k TFlow_{k,i} + R_i \qquad (3.1a)$$

if this river is not a head section, or

$$\mathrm{FFlow_i} = \mathrm{PFlow_i} \bullet \frac{\mathrm{Thrd_i}}{\mathrm{Area_i}} \qquad (3.1b)$$

if this river is a head section, where,

i = a river section's ID (=FNode#),

j = a river section's TNode#,

k = the ID of river sections that have node i as their To-Nodes,

$FFlow_i$ = flow rate at From-Node of river section *i* (m^3/s),

$TFlow_{k,i}$ = flow rate at To-Node (i) of river section *k* (m^3/s),

R_i = source term (pumping or recharging) at node *i* (m^3/s),

$Thrd_i$= threshold defining a river starting point used in basin delineation,

$Area_i$= area of the subwatershed that contains river section *i* (m^2).

The flow rate at the To-Node is given by:

$$TFlow_{i,j} = FFlow_i * C_i + PFlow_i - DFlow_i \qquad (3.2a)$$

if this river line is not a head section, and

$$TFlow_{i,j} = PFlow_i - DFlow_i \qquad (3.2b)$$

if this river line is a head section, where,

i = river section ID,

C_i = river sectional flow gain/loss coefficient of river section *i* (1/m),

$PFlow_i$ = runoff contribution of river basin polygon *i*, that contains river section *i* (m^3/s),

$DFlow_i$ = flow diversion on river section i (m^3/s).

Figures 3.12a and 3.12b show the program flow chart of the module that simulates the water movement between river sections. The center-piece of the algorithm is a stack used to keep track of the stream lines that the model has traveled, so the algorithm can be called a stack-based stream network analysis algorithm. The logic of the algorithm is explained below.

It can be seen from the program flow chart (Figures 3.12a and 3.12b) that, for a given river section, the program starts with searching upstream to find the stream sections flowing into it. There are three possible outcomes of the search.

(1) If no upstream section is found, and the section is not the head section, then a mistake must have occurred either in the programming procedure or in the

stream network formulation. The program will exit and print an error message for program debugging.

(2) If one or more upstream sections are found, at least one of them is not yet simulated, the current river object (identified by its ID) is pushed into the stack and the program moves upstream to one of the river objects not yet simulated. This procedure is repeated until a head section is found. Once a head section is identified, the program searches through the subwatershed polygon attribute table (PFTAB) to identify the subwatershed containing the head section. If none is found, the program exits and prints an error message. Otherwise, the program issues a request to the flow simulation module defined on the subwatershed polygon to get PFlow. Equations 3.1b, and 3.2 are used to compute the FFlow and TFlow of the river object. Once this is done, the algorithm moves to the next downstream river section by issuing a pop request to the stack. At this downstream section, the algorithm again searches upstream and checks the if all the upstream sections have been simulated. Based on the outcome of this search and test, the program decides if it can simulate water movement on the current river object or whether it must push this stream object into the stack and move upstream.

(3) If one, or more than one upstream sections are found, and all have their flow simulations completed (indicated by state value, isdone=1), the river flow is simulated at the current section. The program searches for the subwatershed containing the current river section. One and only one basin should be found. If not, the program prints out an error message and exits. Otherwise, the program issues a request to the subwatershed polygon to get the PFlow and uses Equations 3.1a and 3.2 to compute FFlow and TFlow.

The procedure is repeated until the outlet section is reached and its flow is simulated. This procedure is illustrated in the flowchart shown in Figures 3.12a and 3.12b.

The program is designed in such a way that it allows the model to be applied on any user defined portion of a river basin. The program can also be used to simulate water flow in a region having more than one river network.

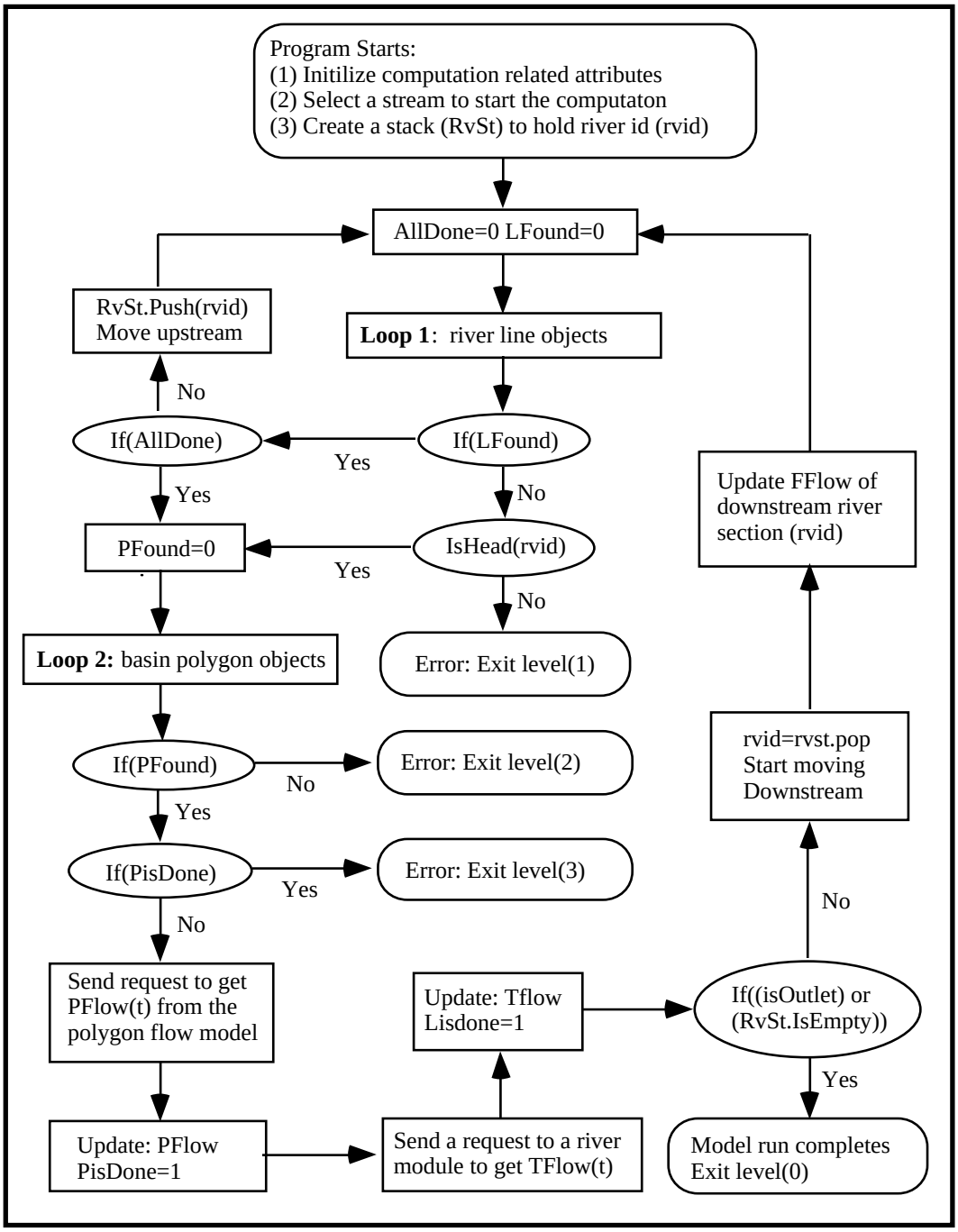


Figure 3.12a. The main loop of algorithm simulating water movement within and between subwatersheds

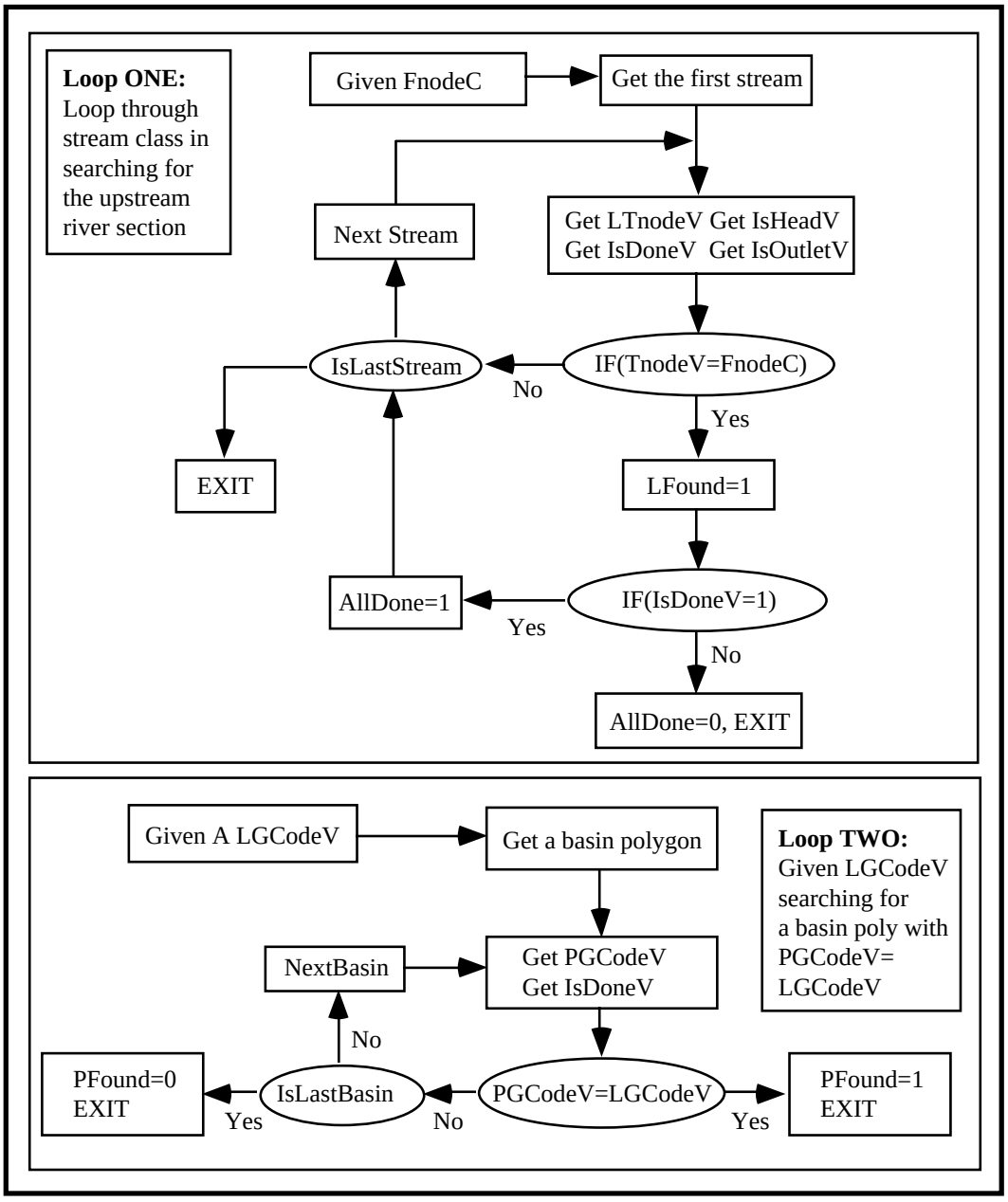


Figure 3.12b. Subloops of the algorithm simulating water-movement within and between subwatersheds

As an example to show how inter-subwatershed water movement is simulated on a river network, line section 25 (be referred to as S25 here after) in Figure 3.13 is selected to start a simulation run. The sequence by which the simulation model is applied to each river section on the stream network is described below.

From S25, realizing its upstream section S27 has not been computed yet, the program pushes S25 into the stack and moves upstream to S27. Because S27 is a head section, the program gets PFlow from the subwatershed containing S27, computes FFlow and TFlow for S27, then issues a pop request to the stack to get S25 and moves downstream to S25. On S25, it finds that S33, another upstream section of S25, has not been simulated. The program again pushes S25 into the stack and moves upstream to S33, gets the PFlow from the polygon containing S33, and moves downstream to S25. This time, the program finds that all its upstream sections (S27, S23) have been simulated and it is time to simulate water movement from the From-Node to To-Node of S25. The model will then activate a module that simulates water movement between the From-Node and To-Node of a river section to simulate the water movement on section S25. After this, the program sends a pop request to the stack to get S18 and moves downstream to S18. This procedure is repeated until the outlet section is reached and simulated. Figure 3.14 gives the sequence by which the program visits and simulates river sections on the river network shown in Figure 3.13.

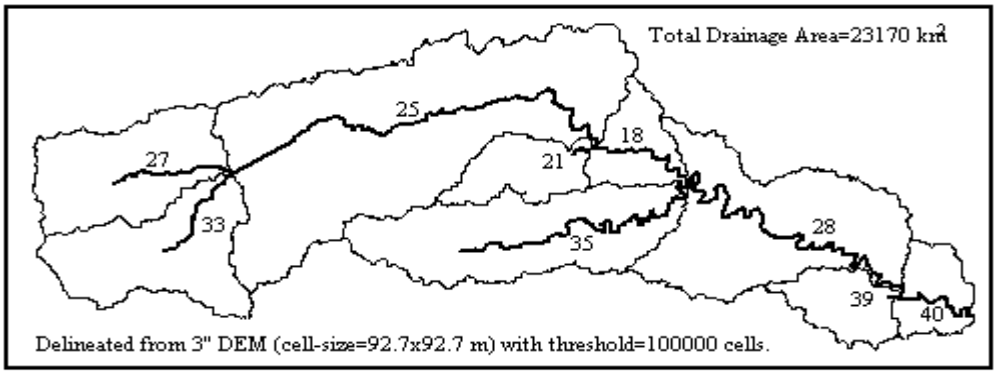


Figure 3.13. The Guadalupe River Basin

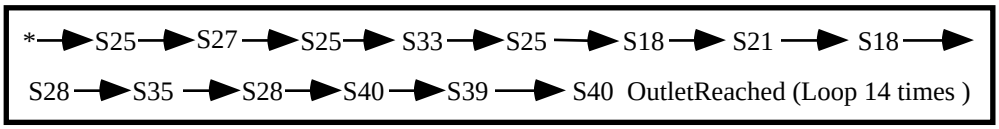


Figure 3.14. Program travel path given by the stack-based algorithm

3.4.2. Simulating Water Movement within a River Section

This section describes how water movement from the From-Node to the To-Node on a river section is simulated. As discussed at the beginning of this section, this simulation model is designed in such a way that by changing the modules used to compute PFlow, FFlow, and TFlow (element functions), the model can be used to simulate different hydrologic processes without changing the stack-based stream network analysis procedure.

In this map-based surface water flow simulation model, four flow routing modules are available to simulate water movement between the From-Node and the To-Node. These four modules use (1) a response function method, (2) a two-step function method, (3) dam/reservoir model combined with the two-step function method, and (4) a Muskingum-Cunge method. All of these four modules

are constructed based on the principle of continuity. Figure 3.15 shows the logic of the program used for river routing. IMax in Figure 3.15 stands for the maximum time step interval allowed before the Muskingum method becomes unstable. HasDam, HasResp, and Muskingum are the states of a river line object given in Table 3.2. As can be seen from this figure, the decision of selecting one specific simulation module (out of four) is made based on the river features at run-time. The theory and programming methodology used to construct each of these four modules are discussed below:

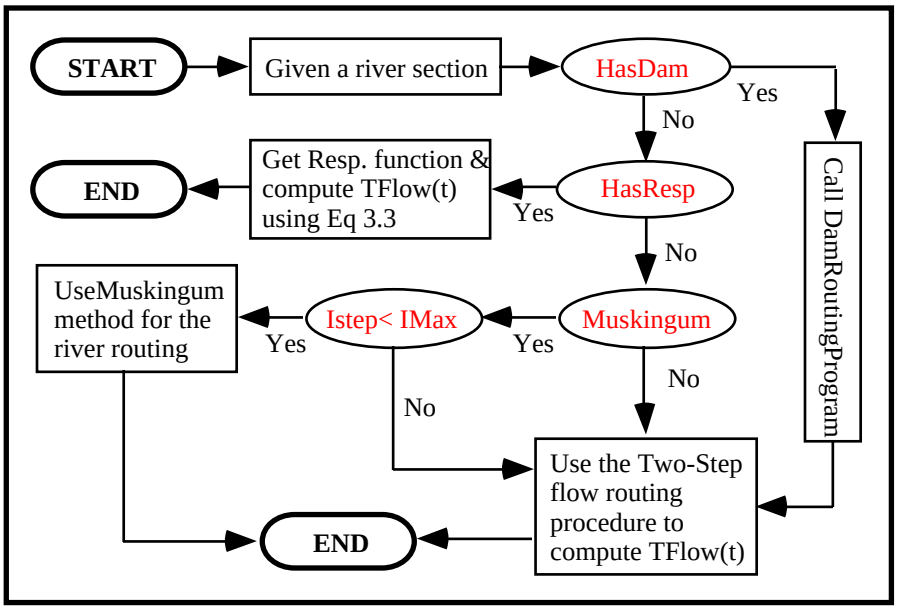


Figure 3.15. Program flow chart of river section flow routing module

(1) The Response Function-Based Flow Routing Module

The state, HasResp, of a river line object, indicates if a user-supplied response function (Maidment, 1993) is available for the river section. HasResp=0 indicates no response function is available and HasResp=k (k≠0) indicates a user-supplied response function is available and the response function has k

components. When a river section has a response function that takes the water flow time-series at the From-Node as an input and produces the flow time-series at the To-Node as an outputs, this response function is used to simulate water movement between the From-Node and To-Node of the river section. The response-function-based flow routing module is constructed using the following equation:

$$TFlow_{i,j}^{t} = \sum_{k=0}^{min(t,N)} FFlow_{i}^{t-k} \cdot U_{i}^{k} + PFlow_{i}^{t} - DFlow_{i}^{t} - Loss_{i}^{t} \qquad (3.3)$$

where,

$TFlow_{i,j}^{t}$ = TFlow of river section i (with j as its To-Node), at time step t (m^3/s),

$FFlow_{i}^{t-k}$ = FFlow of river section i at time step t-k (m^3/s),

U_{i}^{k} = the response function of N elements for river section i,

$PFlow_{i}^{t}$ = PFlow of subwatershed i at time step t (m^3/s),

$DFlow_{i}^{t}$ = flow diversion of river section i at time step t (m^3/s),

$Loss_{i}^{t}$ = stream water loss of river section i at time step t (m^3/s).

The water loss in a river section is computed using the following equation:

$$Loss_{i}^{t} = FFlow_{i}^{t} \cdot Length_{i} \cdot LossC_{i} \qquad (3.4)$$

where, $Length_{i}$ = the length of river section i, (m),

$LossC_{i}$ = loss coefficient of river section i, (1/m), a river line attribute that can be used as a model calibration parameter.

(2) The Two-Step Flow Routing Module

This flow routing model is a special case of the response function flow routing model. This module is used when a user-supplied response function is not available and when the use of Muskingum method is not computationally efficient, i.e., when the size of simulation model time step is much greater than the average travel time through the reach, $\Delta t >> \frac{L_i}{v_i}$. Given below is the equation used for the two-step flow routing module:

$$TFlow_{i,j}^{t} = FFlow_{i}^{t-1} \cdot Llag_{i} + FFlow_{i}^{t} \cdot (1 - Llag_{i}) + PFlow_{i}^{t} - DFlow_{i}^{t} - Loss_{i}^{t} \qquad (3.5)$$

where, $Llag_i$ = The normalized time lag between From-Node and To-Node of river section i,

$$Llag_{i} = \frac{L_i / V_i}{\Delta t}, \qquad (3.6)$$

L_i = the length of river section i (m),

V_i = the average flow velocity on river section i (m/s),

Δt = the size of simulation time step.

As it can be seen from Equation 3.5, the two-step flow routing module is based on the principle of continuity of water flow and guarantees that the water mass will be preserved.

(3) Dam and the Two-Step Flow Routing Module

The state, HasDam, of a river line object, indicates if dams (or reservoirs) exist on the river section. HasDam=0 indicates there is no dam on the river section. HasDam=k, (k≠0) indicates there is at least one dam located on the river section, and the value k is also the ID number of the first dam on the section. When one or more dams exist on a river section, the reservoir routing procedure (DAMRT.ave) is called to simulated the effects of reservoirs on the river flow. Whenever reservoirs exist on a river section, the two-step flow routing module is used to simulate the water movement on the river segments between the dams and From-Node and To-Node of the river section. The dam simulation module is discussed in Section 3.5.1.

(4) Muskingum-Cunge Flow Routing Module

The state, Muskingum, of a river line object, indicates if the Muskingum-Cunge river routing method will be used for the river section. Muskingum=0 indicates the Muskingum method will not be used to simulate water movement on the river section and Muskingum=n (n≠0) indicates that the Muskingum flow routing method will be used if the number of internal loops is less than n. The internal loops are constructed to ensure the computational stability of the Muskingum river flow routing procedure. The internal loop is discussed further below. The Muskingum-Cunge method is introduced in Chapter Two and the flow routing equations derived for the method are reproduced here as Equations 3.7 and 3.8.

$$TFlow_{i,j}^{t} = C_1 \cdot FFlow_i^t + C_2 \cdot FFlow_i^{t-1} + C_3 \cdot TFlow_{i,j}^{t-1} + C_4 \qquad (3.7)$$

where,

$$C_1 = \frac{\Delta t - 2KX}{2K(1-X) + \Delta t} \quad (3.8a)$$

$$C_2 = \frac{\Delta t + 2KX}{2K(1-X) + \Delta t} \quad (3.8b)$$

$$C_3 = \frac{2K(1-X) - \Delta t}{2K(1-X) + \Delta t} \quad (3.8c)$$

$$C_4 = \frac{PFlow_i^t - DFlow_i^t - Loss_i^t}{2K(1-X) + \Delta t}, \quad (3.8d)$$

$K = \frac{\Delta x}{\bar{c}}$, K is a storage constant [T], (3.8e)

$$X = \frac{1}{2} - \frac{Avg(TFlow, FFlow)}{2\bar{c}\bar{B}S_e\Delta X} \quad (3.8f)$$

X = a weighting factor showing the relative importance that FFlow and TFlow have on a river section's storage, $(0 \le X \le 0.5)$, where 0 indicates pure storage and 0.5, indicates pure translation,

$\bar{c}$ = kinematic wave velocity (m/s),

$\bar{B}$ = cross-sectional top width associated with average of TFlow and FFlow,

S_e = the energy slope,

Δx = the length of a river section.

To ensure the stability of the flow routing, C_3 needs to be greater than zero. From equation 3.7c, it can be seen that in order for $C_3 \ge 0$, we need to have $\Delta t \le 2K(1-X)$. When the simulation time step Δt does not satisfy this relationship, the time step needs to be subdivided into smaller steps. In the simulation model, $C_3 \ge 0$ is achieved by subdividing a simulation model time

step into multiple n internal time steps so that $\Delta t_i = \frac{\Delta t}{n} \le 2K(1-X)$. When n is too large, the round-off errors accumulated by the internal loop may outweigh the benefits brought by using the Muskingum-Cunge method. In this event, the simulation model switches to use the two-step function flow routing module (Figure 3.15).

3.4.3. Simulating Water Movement within a Subwatershed

In this map-based surface water flow simulation model, PFlow(t) represents local runoff generated in a subwatershed. Several methods are available to calculate PFlow(t) from precipitation observations, e.g. by spatially interpolating the precipitation to the center point of each subwatershed and then converting the precipitation to PFlow(t). In the Niger River Basin simulation model, PFlow(t) is constructed from the soil-moisture surplus time-series produced using a simple bucket model. As the input data sets for the soil-moisture balance model, monthly time-series tables of precipitation and potential evaporation for the period July 1983 to December 1990 were estimated at each point on a regular mesh of 0.5 degree cells covering the study area. This computational mesh was selected because long term mean monthly estimates of rainfall and temperature at these points were available from C.J. Willmott at the University of Delaware. As discussed in Chapter Two, a convolution procedure (Olivera and Maidment, 1996) is used to convert soil-moisture surplus to local runoff (SurpF(t)-> PFlow(t)).

Using this method, PFlow(t) is simulated with two flow components. One is SFlow(t) used to simulate the overland flow and the other one is OFlow(t) used to simulate the subsurface water flow. The overland flow is estimated by applying the concept of unit hydrograph to the soil surplus, SurpF(t). The

equations that the subwatershed to river water movement simulation module uses are given below:

$$SFlow_i^t = \sum_{k=0}^{\min(t,N)} SurpF_i^{t-k}(1-\alpha_i)\cdot U_i^k \qquad (3.9)$$

where,

$SFlow_i^t$ = local flow contribution (through surface) of subwatershed i at time step t (m^3/s),

$SurpF_i^{t-k}$ = soil moisture surplus of subwatershed i at time step t-k (m^3/s),

U_i^k = k-th component of the response function of $PFlow_i^t$ on $SurpF_i^t$,

α_i = the fraction of surplus that goes to subsurface, $(0 \le \alpha_i \le 1)$,

N = total number of components in the response function U_i^k.

The response function of used in this study is given below:

$$U_i^k = \frac{1}{2k\sqrt{\pi D_i \frac{kv_i}{T_i}}} \exp\left(-\frac{(1-\frac{kv_i}{T_i})^2}{4D_i\frac{kv_i}{T_i}}\right) \quad k=1,2,3....N \qquad (3.10)$$

where,

k =1,2,3...N, the index of components in the response function,

D_i= dispersion coefficient for subwatershed i, Dispersion coefficient is used to measure the degree of the overland water spreading through time.

V_i = average overland flow velocity for subwatershed i, (m/s),

T_i = average overland flow time for subwatershed i (s).

The values of these parameters for a subwatershed can be estimated either through hydrologic analysis of the subwatershed or through the optimization module to be discussed later in this chapter.

The subsurface water flow component of PFlow(t) is considered to be going through an imaginary underground reservoir. The flow through the reservoir can be simulated using a linear reservoir model. Equations used to perform flow routing through a linear reservoir are given below:

$$OFlow_i^t = S_i^{t-1} / K_i \qquad (t = 1, 2, 3, ...) \tag{3.11}$$

$$S_i^t = S_i^{t-1} + (SurpF_i^t \cdot \alpha_i - OFlow_i^t) \cdot \Delta t \tag{3.12}$$

where,

$OFlow_i^t$ = the subsurface flow component at time step *t*, on polygon i (m^3/s)

S_i^t = storage of the under-ground-reservoir at time step *t*, on polygon *i* (m^3),

K_i = the linear reservoir constant representing the average residence time of water in the reservoir (s).

After these two components are computed, the local flow contribution of subwatershed *i* at time step *t* is computed using Equation 3.13.

$$PFlow_i^t = SFlow_i^t + OFlow_i^t \tag{3.13}$$

3.5. OTHER SIMULATION MODEL OBJECTS

This section describes three other classes of objects used in the simulation model, dam/reservoir object, flow-check object, and flow-diversion object. All these objects appear as a points on a point coverage and are located on a river line object. The point location on the river line is dynamically calculated from a user specified cursor location. The algorithm that performs the dynamic line segmentation is described in Section 3.6.4.

3.5.1. The Dam Objects

The dam/reservoir objects are designed to simulate the effects of dams and reservoirs on river flows. The states of a dam/reservoir object are given in Table 3.3. A utility program is available to construct dam/reservoir object directly and interactively from a river map. For each dam object created, a new record is added to the dam feature attribute table (FTAB) and a time-series table is created to hold time-series data sets for reservoir routing. The states of a dam object are listed in Table 3.3a and the fields in the dam routing time-series table are given in Table 3.3b. The graphic point on the dam coverage is connected to the dam's FTAB through the shape field in the FTAB. The connection to its associated time-series table is established through the DamId and the name of the time-series table.

During a flow simulation, the presence of a dam/reservoir object is detected by the simulation model through the return value of the HasDam state of a river section.

HasDam=0 indicates that no dam is located on the river section. The return of a non-zero value of HasDam indicates that there is at least one dam on

the river section and the non-zero value is the Dam-ID of the first dam located on the river section.

Table 3.3a. The Attributes of a Dam/Reservoir Object

State	**Function (What the attribute represents)**
Shape	Pointer pointing to the map location of the object
Damid	The ID of a dam/reservoir object, Damid is also a pointer pointing to the time-series table associated with the reservoir
DamName	The name of a dam/reservoir
Capacity	Capacity of a reservoir (m^3)
Area	The water surface area of the reservoir when storage=capacity (m^2)
Upst	The active storage of the dam/reservoir (m^3)
DeadSt	The dead storage of the dam/reservoir(m^3)
Evt	The evaporation rate of the reservoir (mm/day)
Pdam	Damid of the upstream dam located on the same section, 0=No upstream dam
Ndam	Damid of the dam downstream of the dam on the same stream section, 0=No dam located downstream
Storage0	Initial storage of the reservoir (m^3)
Area0	Initial water surface area of the reservoir (m^2)

Table 3.3b. The Fields of a Dam-Routing Time-Series Table

FieldName	**Function (What the attribute represents)**
Time	Simulation time steps
Inflow	Inflow time-series of the reservoir (m^3/s)
DmdFact	Demand factor of each simulation time step
WithDraw	Water withdraw of each time step
NetEvp	Net evaporation of each time step (mm/day)
Sarea	Surface area of each simulation time step (m^2/s)
Spill	Discharge (including water used for power generation) of each time step (m^3/s)
Storage	Storage at the end of each simulation time step (m^3/s)

In the process of a river flow routing, if a reservoir is detected on a river section, the river routing program will call the dam simulation module DAMRT and pass the Dam-ID and FFlow(t) to DAMRT. The DAMRT module will then simulate the behavior of the reservoir. The simulation is based on the information

passed to it from the river-routing program, the states of the dam object, and the reservoir operating rules. The results of the simulation are stored in a time-series database table associated with the dam, of which, discharge time-series (spill-vector) is passed back to the river routing program to be used as FFlow(t). If more than one dam exist on a river section, the output of the first dam becomes the input of next dam. This loop will continue until the last dam of the river section is reached and simulated. The output of the last dam is then returned to the river routing program to be used as FFlow(t).

The module DAMRT, that simulates a reservoir water balance is constructed based on equation 3.14 (Chow *et al.*, 1987):

$$S^t = S^{t-1} + (I^t - Yd^t - A^t e^t - Q^t)\Delta t \qquad (3.14)$$

where,

I^t = inflow in time step t (m^3/s),

S^t = reservoir storage at time step t (m^3),

Yd^t = withdrawal at time step t, given by the reservoir operation rule (m^3/s),

$A^t e^t$ = evaporation loss at time step t (m^3/s),

e^t = evaporation loss rate at time step t (m/s),

A^t = reservoir water surface area (m^2).

The reservoir water surface area, A^t, is estimated using the equation (Woelke, 1985):

$$A^t = (S^{t-1})^{0.72} \qquad (3.14a)$$

where, Q^t = reservoir water release to downstream at time step t (m^3/s).

Reservoir release is given by the reservoir's operating rule subject to the following constraints:

$$Q^t \geq \frac{S^{t-1}}{\Delta t} + (I^t - Yd^t - A^t e^t) - \frac{S_c}{\Delta t} \qquad (3.15a)$$

$$Q^t \leq \frac{S^{t-1}}{\Delta t} + (I^t - Yd^t - A^t e^t) - \frac{S_d}{\Delta t} \qquad (3.15b)$$

where, S_c = the reservoir storage capacity (m^3),

S_d = the reservoir dead storage (m^3).

Relation 3.15a ensures that at any time step, the reservoir storage will not be greater than the reservoir capacity, and relation 3.15b ensures none of the water in the dead storage goes into release.

Inflow to a reservoir is computed differently based on the location of the reservoir. When a reservoir is located at a non-head river section and is the first on the river section, Equation 3.16a is used to compute the inflow.

$$I^t = FFlow^t(1 - Llag) + FFlow^{t-1} \cdot Llag + PFlow^t \cdot \frac{\Delta A}{A} \qquad (3.16a)$$

where,

$Llag = \frac{(XL/v)}{\Delta t}$ = fractional time it takes for water to travel from the From-Node to the location of the dam,

XL = the distance between the From-Node and dam location (m),

v = river flow velocity (m/s),

Δt = the size of time step of the simulation (s),

ΔA = regional area between FNode and the reservoir location (m^2). When ΔA is not given, it can be estimated using Equation 3.16b:

$$\Delta A = \frac{XL}{TL} \cdot A \qquad (3.16b)$$

A = the total area of the subwatershed where the reservoir is located,

TL = total length of the river section.

When a reservoir is located at a head section and is the first on the river section, Equation 3.16c is used to compute the inflow.

$$I^t = \left[PFlow^t \cdot (1 - Llag) + PFlow^{t-1} \cdot Llag\right] \cdot \left(\frac{\Delta A_{thrd}}{A}\right) + PFlow^t \cdot \frac{\Delta A}{A} \qquad (3.16c)$$

where, ΔA_{thrd} = threshold area that initializes a river used in watershed delineation procedure.

Comparing Equation 3.16c with 3.16a, it can be seen that in the head river section, $FFlow^t = PFlow^t \cdot \left(\frac{\Delta A_{thrd}}{A}\right)$. When a reservoir is not the first reservoir on a river section, its inflow is computed using equation 3.16d.

$$I_i^t = \left[Q_{i-1}^t \cdot (1 - Llag) + Q_{i-1}^{t-1} \cdot Llag\right] + PFlow^t \cdot \frac{\Delta A}{A} \qquad (3.16d)$$

where,

i = sequential numerical index of the dams in a river section with i=1 for the first dam on the river section. i=2,3,4..,

$Llag = \frac{(XL/v)}{\Delta t}$ the fractional time it takes for water to travel from dam i-1 to i,

XL = travel distance along the river line from dam i-1 to i,

Q_{i-1}^t = water release of reservoir i-1 at time step t. (m^3/s).

It can be seen from the equations above that dam simulation module uses Equations 3.14 and 3.15 to perform the reservoir water balance computation and uses the two-step flow routing module to simulate the flow in the river segments between the dams. More complex reservoir operating rules such as the ones introduced in the works of Loucks (Loucks, 1981) could also be included in this dam/reservoir simulation model.

3.5.2. The Flow-Check Point Objects

The flow-check objects are created at the locations where stream flow estimations are desired because the simulation model produces the flow estimates only at the From-Node and To-Node of each river section. A utility program is written to allow the interactive construction of these objects on a river network. Table 3.4 lists the attributes of a flow-check point object. The algorithm used to interpolate flow rate to a flow-check point is discussed in Section 3.6.4.

Table 3.4. Attributes of a Flow-Check Point Object

State	Function (What the attribute represents)
Shape	Pointer pointing to the map location of the object
Pointid	Identification number of a flow-check point
FFlow	FFlow of the river line object on which a flow-check point is located
TFlow	TFlow of the river line object that a flow-check point is located
IFlow	Flow interpolated at a flow-check point
Oline	A flag specifying if a flow-check point is located on a river (Oline=0 indicates the flowchk point is on the river line, Oline=1 indicates the point is not on the river line but within the simulated area, and Oline=-1 indicates the point is out of the area)
Pcntage	between the distance of a Flow-Check point to From-Node and the length of the river section.
Length	The length of the river section where a flow-check point is located

3.5.3. The Flow-Diversion Point Objects

When a flow diversion point has a constant flow rate, it is incorporated into the river object associated with the point. The diversion rate is put into the attribute DFlow of a river object. When a flow diversion point has a time-varying flow rate, it can be simulated using the reservoir object with constant storage and no evaporation loss. These objects are detected and simulated by the river flow routing module.

3.6. UTILITY PROGRAMS AND POST-PROCESSORS

The utility programs of this map-based simulation model are constructed to allow interactive model modification and make easy result presentations possible. This section describes the designs of these interactive model modification and result presentation programs.

3.6.1. Construction of a Sub-Model

When simulating surface water flow process on a large river basin, it is sometimes necessary to isolate a portion of the river basin for more detailed study. For this purpose, it is desirable to clip a portion of the river basin from the main model to form a sub-model. When a simulation model is constructed in the traditional means, constructing such a sub-model requires an effort comparable to that used to construct the original basin model. In this map-based simulation model, however, a sub-model can be constructed interactively from the existing basin model in a 'real-time' operating style, i.e. when a portion of the map is selected and copied, a new model of the selected region is created. The procedure to create a sub-model is listed below:

(1) select the maps necessary to create the model by making their corresponding themes active in an ArcView's view window,

(2) select the features (polygons, lines, or points) from each map that fall into the study region of the sub-model,

(3) run the sub-model construction utility program (CLIPMDL.utl) to create the sub-model.

Because the model is constructed based on the concept of object-oriented programming, the newly created sub-model inherits all the features of the original model. Therefore, once constructed, all the programs applied to the original model can be applied to the sub-model. Figure 3.17 in Section 3.6.2 shows an example of a sub-model. The logic of the program that enables interactive sub-model creation is explained below.

It is known that for each feature object on a geographic map there is a record in the relational database table containing the states of the object and the model programs have the ability to access both the map feature and its associated

record. In procedure (1), at the same time when each map is selected, its related database table is also selected, and in procedure (2), when features on these maps are selected, their corresponding records in the database tables are also selected.

The functions of the sub-model construction program [CLIPMDL.utl] are (1) to make one copy of map-templates (class) for each user-selected map, (2) to loop through each selected database table and copy the selected records to its newly created map-template, and (3) to create an alias for the newly-created maps so that they are the same as those for the maps in the original models. The reason that newly created maps keep the same alias as those of its original maps is that the maps are referenced in the model programs by their alias. Programs identify maps by their alias rather than their real file name, and if the maps of the sub-model have the same alias as those of original maps, all the programs used in the original model can be applied to the sub-model without any modifications.

The sub-model construction utility program also identifies the sources of the river network so that inputs related to those streams can either be inherited from the original model simulation results or supplied by the user. A source stream is defined as a river section whose inflow streams in the original model are cut off by the clipping procedure.

3.6.2. Optimization Algorithms

This section describes the optimization algorithms developed for the purpose of simulation model calibration. Two optimization algorithms, multi-directional and interactive, are developed in this research for the purpose of model calibration. Both algorithms require a user to provide, on a space defined by the optimization variables, a value range for each variable to be calibrated. The interactive model usually requires multiple runs to get an optimal solution and for each optimization process, a new solution space based on previous runs needs to

be defined. The second method, multiple directional optimization requires a user to define only the solution space at the beginning of a calibration process. After the solution space is defined, the optimization model applies a bisection algorithm on each dimension to search for an optimal solution set. The concepts of both optimization modules are discussed below in Sections 3.6.2.1. and 3.6.2.2.

3.6.2.1. The Interactive Optimization Algorithm

The purpose of this interactive optimization algorithm (IOA) is to calibrate the model parameters for this map-based simulation model. The optimized parameters are defined as a set of parameters that produce a best-fit between a simulated and observed river flow time-series at a given location. Two standards (match standard) used by the algorithm to evaluate the extent of fit are: sum of mass discrepancies (SMD) and sum of root mean square errors (RMSE). A best-fit time-series can be defined as the time-series that minimizes RMSE and produces zero SMD. SMD function is used to ensure that the average flow rate produced by the simulation model equals that of the observed in the same period of time. RMSE function is used to pursue the best match between the simulated flow time-series and observed flow time-series. Since the model is designed in such a way that the optimization algorithm is independent of the match standards used, a user of the model can add other standards to achieve better matches on the low flow and peak flow period. Mathematically, the sum of mass discrepancies can expressed as:

$$SMD = \frac{\sum_{t=1}^{n}(O^t - F^t)}{\sum_{t=1}^{n} O^t} \qquad (3.17)$$

The sum of root mean square error can be expressed as:

$$RMSE = \frac{\sqrt{\sum_{t=1}^{n}(O^t - F^t)^2}}{\sum_{t=1}^{n} O^t} \tag{3.18}$$

where, O^t = observed flow rate at time step t at a flow gage station,

F^t = simulated flow rate at time step t at the same flow gage station. The function F^t can be written as:

$$F^t = F(x_1, x_2, x_3, ...x_n) \tag{3.19}$$

where, $x_1, x_2, x_3, ...x_n$ = model parameters. The parameter optimization problem becomes the problem of finding a parameter set $\vec{P} = \{x_1, x_2, x_3, ...x_n\}$ that satisfies:

$$SMD = \frac{\sum_{t=1}^{n}(O^t - F^t)}{\sum_{t=1}^{n} O^t} = SMD(x_1, x_2, x_3, ...x_n) = 0 \tag{3.17a}$$

and minimizes:

$$RMSE = \frac{\sqrt{\sum_{t=1}^{n}(O^t - F^t)^2}}{\sum_{t=1}^{n} O^t} = \text{RMSE}(x_1, x_2, x_3, \ldots x_n) \qquad (3.18a)$$

As an example to show how the interactive optimization algorithm works, the procedure of optimizing two variables, X_1 and X_2, for a simulation model based on the standards imposed in Equations 3.17 and 3.18 is described below.

The algorithm assumes that the lower and upper limits of X_1 and X_2 are known or could be reasonably estimated. Let the lower and upper limit for X_1 and X_2 be designated as x_{1min}, x_{1max}, x_{2min}, and x_{2max}, respectively. The optimization problem is equivalent to finding the X_1, X_2 combination from the parameter space $\{X_1,X_2\}$ that gives a minimum SMD and RMSE (Figure 3.16). The algorithm first discretizes each variable (represented as one dimension in the parameter space) into a user specified range. In this example, 6 steps are specified for parameter X_1 and 5 for X_2. As a result, 30 X_1~X_2 parameter combinations are formed. The interactive optimization algorithm then uses these parameter sets to run the simulation model 30 times, and computes the RMSE and SMD based on each simulation result. The parameter set that returns the minimum RMSE or SMD is manually selected as the first estimate. A finer step size can now be used to form another grid around the first estimate to form another parameter set in search for a better fit. The procedure can be repeated several times until an optimal is found.

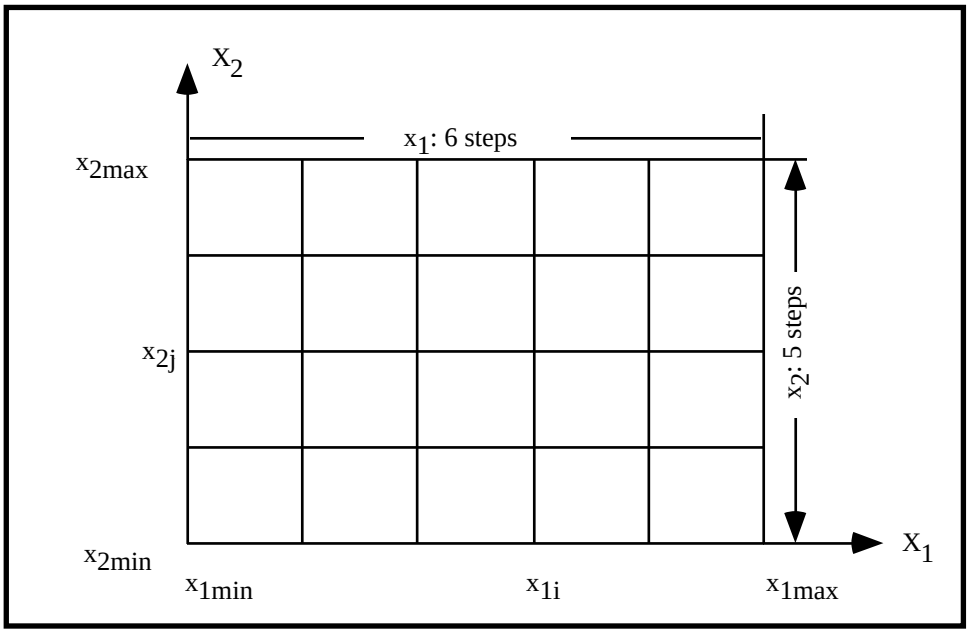


Figure 3.16. Problem solution space X_1~X_2

The procedures of applying this interactive optimization model to the region above the Koulikoro flow-gauging station in the Niger River Basin is described below to show the technical details of the optimization model application. Figure 3.17. shows the region whose parameters are to be optimized.

The region above the Koulikoro flow-gauging station consists of 10 subwatersheds with a total drainage area of 120,000 km^2. The flow-gauging station is located at a distance 0.814 percent (measured from the From-Node of the arc) of the total length on the outlet river section. Monthly runoff records of 90 months from July 1983 to December 1990 at the Koulikoro flow-gauging station are processed and used. The input data sets for the map-based simulation model are the time-series of soil-moisture surplus defined on each subwatershed polygon. The time-series of soil-moisture surplus are produced using the soil-water balance model described in Chapter Two.

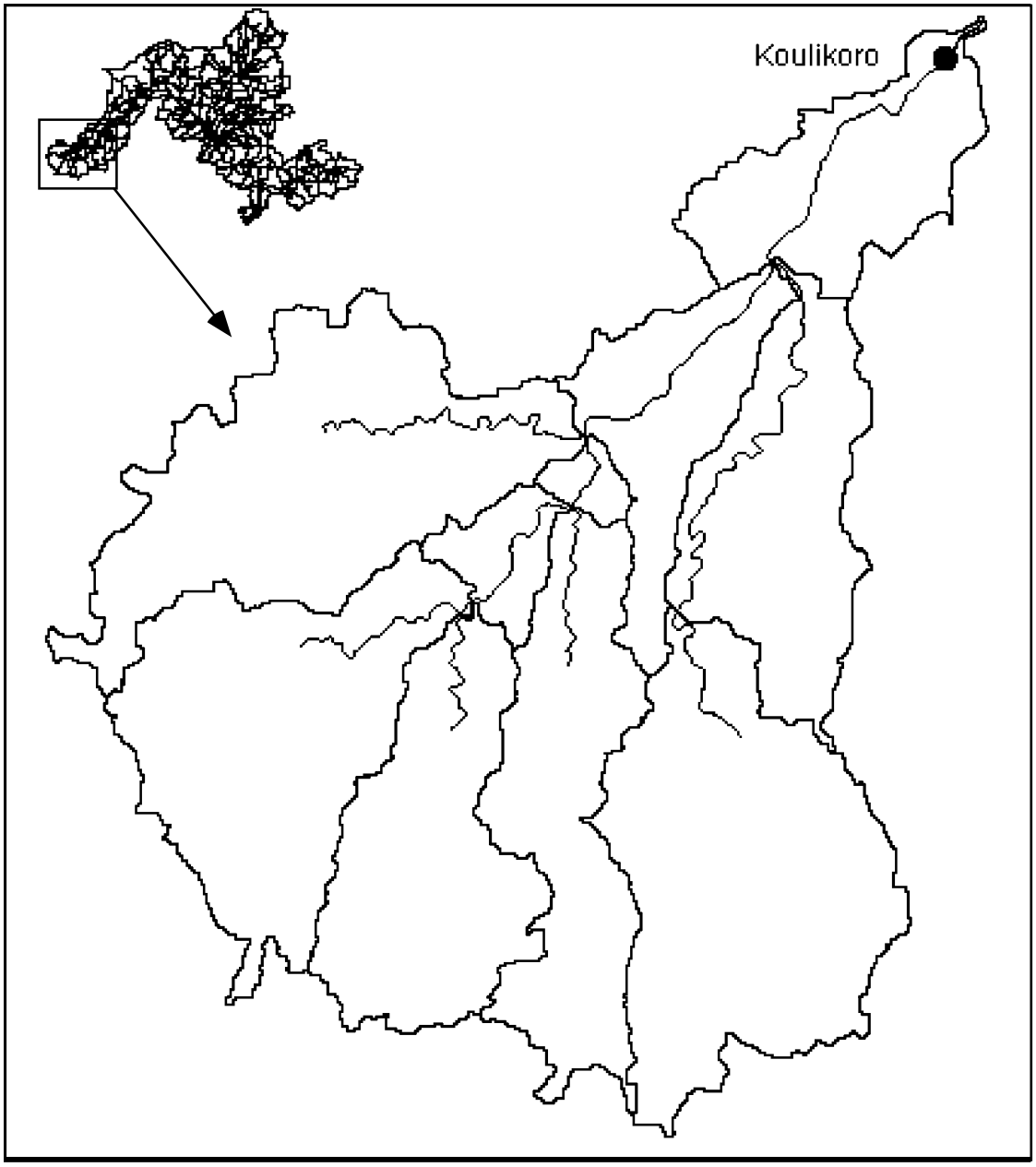


Figure 3.17. The area above the Koulikoro flow-gauging stations

The purpose of applying the optimization model to the region is to find a set of values for the loss-coefficient and flow velocity of each river section so that

the river flow time-series produced by the simulation model at Koulikoro matches that of observed at the location. The match is defined by Equations 3.17 and 3.18.

As an estimate, ranges of the loss-coefficient[1] and average flow velocity on the river are estimated as below:

$$0.00001 \leq \text{lossC} \leq 0.0007 \text{ (1/km)}$$
$$0.2 \leq V \leq 0.55 \text{ (m/s)} \qquad (3.19)$$

The range of each variable is divided into 4 intervals with 5 steps, which creates a testing space of 5x5=25 data sets. After applying the interactive optimization model under such conditions, four charts are generated by the model to show the parameter values that produce the best fit. These four charts are (1) RMSE vs. parameter set (Figure 18a), (2) SMD vs. parameter set (Figure 18b), (3) flow time-series plot of observed vs. simulated when RMSE is the minimum, and flow time-series plot of observed vs. simulated when |SMD| is the minimum. The last two charts are not shown here because similar charts will be produced later in Section 3.6.2.2.

Figures 3.18a and 3.18b show the plots of SMD and RMSE for each parameter sets. It can be seen from the plots that the minimum RMSE can be achieved by adjusting the river velocity while the minimum SMD can be achieved by adjusting the loss-coefficient of the river. In most situations, to have zero mass discrepancy between the simulated time-series and the observed time-series is very important. Figure 3.18b shows that the SMD curve crosses zero under all 5 velocity values used indicating the zero mass discrepancy can be achieved by varying the loss-coefficient alone. Therefore, by selecting a flow velocity that

produces a minimum RMSE followed by adjusting the loss-coefficient of river sections to get a zero mass discrepancy, it is possible to find a set of velocities and loss-coefficient that produce zero SMD while minimizing RMSE.

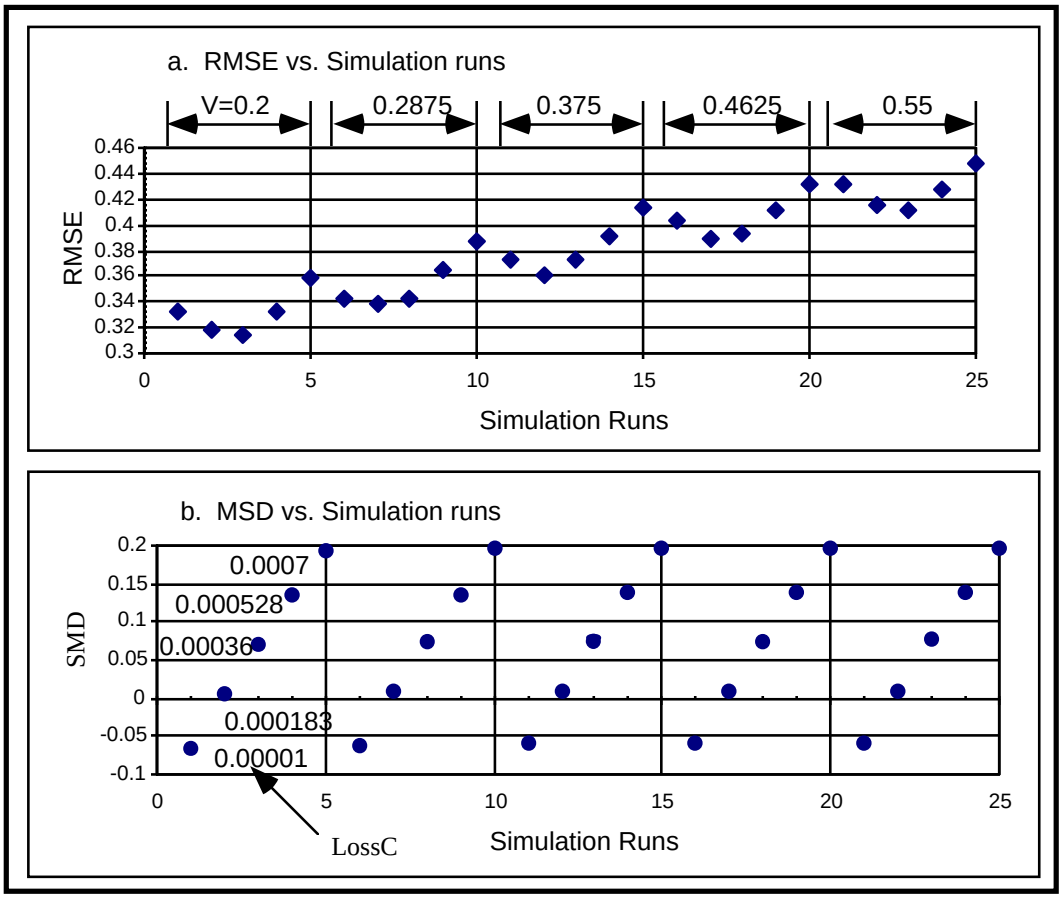


Figure 3.18. Model calibration (SMD and RMSE vs. simulation runs)

Because this optimization run only produced the minimum RMSE and |SMD| out of the predetermined parameter sets, further optimization runs are still needed in order to find the river velocity and loss coefficient combination that produces the minimum RMSE and |SMD| values. As can be seen from Figure

[1] . Loss in a river section is computed by: Loss=RiverLength*LossC*FlowRate while flow loss on

3.18a, the value of RMSE may still go down if river velocity is further reduced. Because for each run, this optimization model requires a user to specify a new range for each model calibration parameter based on the previous run, using this interactive optimization for the simulation model calibration can be a lengthy process.

Another problem one may face when using this interactive optimization model is to decide the ranges of parameters for a new run after an optimization run is done because when the number of calibration parameters is greater than two, the plots like Figure 3.18 will not be available, and without these plots, the patterns of RMSE and SMD variations cannot be easily detected. Because of these problems, this interactive optimization model is never used to calibrate the simulation model in this study. Instead, the optimization model based on a direction set method described in the following section is responsible for all the model parameter calibrations.

The reasons why this interactive optimization model is discussed in this section are (1) this model provides an alternative means of simplifying model calibration procedure when the number of parameters to be calibrated is low and (2) this model produces charts showing the effects that each model parameter has on the changes of RMSE and SMD (Figure 3.18) when the number of parameters is two.

3.6.2.2. Optimization Module Based on a Direction Set Method

Using directional method, the optimization problem presented in Equations 3.17a and 3.18a can be stated as:

---

a subwatershed is computed by: Loss=MeanFlowLength*LossC*PFlow

Given as inputs the vectors $\vec{P}$ and $\vec{n}$, and functions, $RMSE(\vec{P})$, from Equation 3.18a, and $SMD(\vec{P})$, from Equation 3.17a, find a scalar set λ_n that minimizes $RMSE(\vec{P}+\lambda_n\vec{n})$ and satisfies $SMD(\vec{P}+\lambda_n\vec{n})=0$, where $\vec{P}$ is a vector pointing to a point *P* in a *N*-dimensional space and $\vec{n}$ is the unit vector in nth dimension of the *N*-dimensional space. Once the scalar set is found, the new vector $\vec{Q}=\vec{P}+\lambda_n\vec{n}$ is pointing to the point in the n-dimensional space whose coordinates gives the optimal values of the parameter set.

In searching for the optimal vector $\vec{Q}$, this optimization model uses the bisection method (Press *et al.*, 1992) in each dimension to perform successive line minimization. The bisection method is used for its simplicity and reliability, i.e. when a root is contained in a range, the root will not be lost during the iterative root-finding procedure if bisection method is used.

Before the bisection method can be used to search for a root of a function *G(x)=0* (Figure 3.19(A)), an interval $[x_a,x_b]$ containing the root needs to be provided. For a continuous function, whether the interval contains a root can be easily checked by testing the function values at the end points of the interval. If the function values have different signs at the end points, then at least one root is contained in the interval. The concept of the bisection algorithm is simple. Once an interval containing at least one root is given, the algorithm will evaluate the function at both endpoints and the midpoint. The endpoint giving the function value the same sign as that of the function value at the midpoint is then replaced by the midpoint. The procedure is repeated until the function value at the midpoint is sufficiently close to zero. The criteria for convergence is problem

dependent and can usually be given by a user. For our problem of solving equation $SMD(x) = 0$, the convergence criteria are:

$$\left|SMD(x_{mid})\right| < 10^{-5} \qquad (3.20a)$$

or

$$\frac{\left(\left|x_1\right| + \left|x_2\right|\right)}{2} < \varepsilon \approx \sqrt{k} \qquad (3.20b)$$

where, $x1,x2,x_{mid}$ = the endpoints and midpoint of last iteration, respectively,

ε = the convergence criterion given by a user, and

κ = a computer's floating point precision, 10^{-8} for float (single precision) and 10^{-15} for double precision. Similar convergence criteria is used for the procedure that minimizes the RMSE.

Detailed discussion regarding how the convergence criteria should be set can be found in books on numerical methods (e.g., Press *et al.*, 1992).

Using the bisection algorithm to search for the minimum point of a function follows a similar logic to the method used to find a function root. Starting with a given interval $[x_a,x_b]$ and $x_b>x_a$ (Figure 3.19(B)), the program first evaluates the values at the endpoints and the midpoint of the interval. If the function value at the middle point is greater than those at the endpoints, the program will continue on to evaluate the function points at the locations of $x_a+0.25(x_b-x_a)$, $x_a+0.75(x_b-x_a)$, and so on, until a point whose function value is less than at least one endpoint is found, and the point would be used as x_{mid}. To decide the next move, the program evaluates the function value at $x_{tmp}=x_{mid}+dx$, where dx is a small number comparable to ε in Equation 3.20b. If $F(x_{mid}) <$

$F(x_{tmp})$, then x_{mid} is used to replace x_b to start the next iteration, otherwise, x_a will be replaced. The procedure is repeated until the function F(x) converges to its minimum point or until the user specified maximum number of iterations is reached.

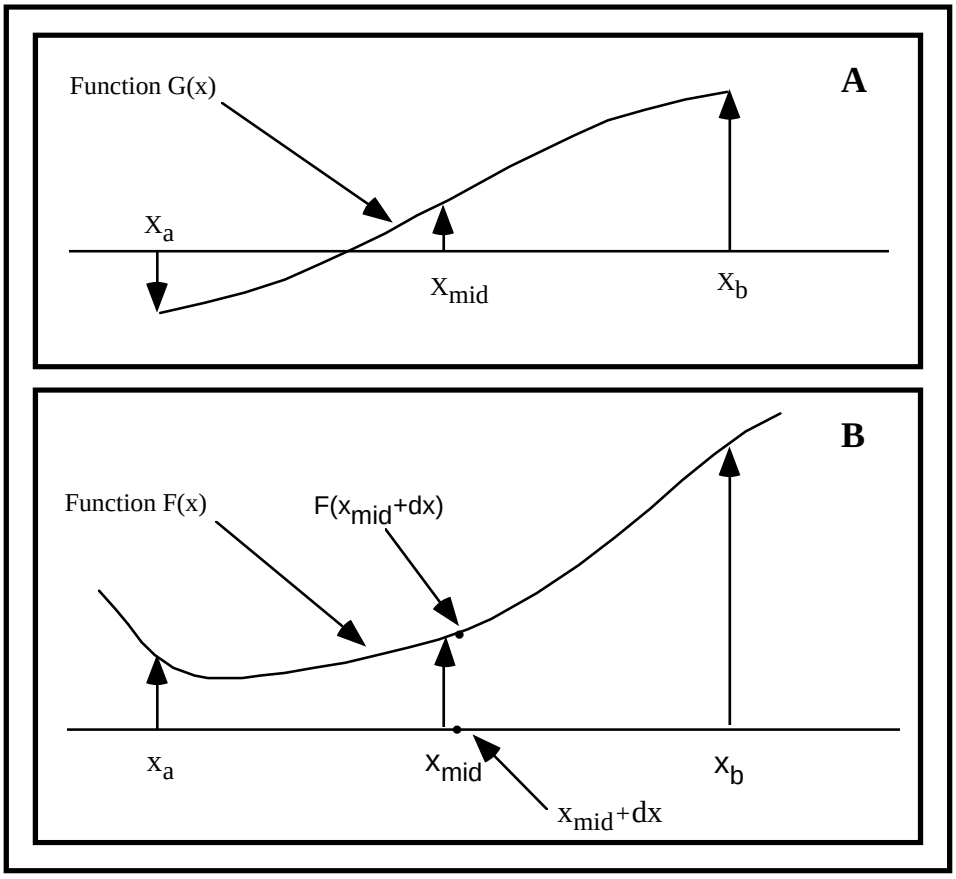


Figure 3.19. Using bisection method to find root and minimum points of a function

3.6.3. Simulation Model Calibration

The optimization models developed in the last section are used in this section to optimize model parameters for the purpose of simulation model calibration.

Figure 3.20. shows the map of the Niger River Basin and the locations of the flow-gauging stations where reasonably reliable observed monthly flow time-series data are available for the time period between July of 1993 and December of 1990. For model calibration purposes, five sub-models, each associated with a flow-gauging station, are created. The optimization model based on the direction set method is applied to each sub-model to optimize the model parameters within its region. Based on river basin characteristic and the simulation model structure, it was decided that six parameters affect the simulated river flow time-series. These six parameters are listed in Table 3.3.

Table 3.5. Simulation Model Parameters To Be Calibrated

	State	Function (What the attribute represents)
1	ToRes	The fraction of a subwatershed water surplus that goes to the subsurface reservoir
2	ResK	Mean residence time of water in a subsurface reservoir [T]
3	VFact	Overland flow velocity (m/s)
4	PlossC	Subwatershed loss coefficient (1/m)
5	Velocity	Flow velocity on a river line (m/s)
6	LossC	Loss coefficient related to a river line (1/m)

After applying the optimization module to the Koulikoro sub-model (Figure 3.17) with all six parameters selected it was found that the parameters ToRes and ResK have little impact on the simulation results, although the combination of ToRes=0.1 and ResK=7 produces slightly better results than other combinations of these parameters. Therefore, ToRes=0.1 and ResK=7 are used throughout the whole model calibration process.

Because both river velocity and overland flow velocity (Vfact) affect flow distributions over the time domain, only one of them is needed for the optimization. For model calibration, the overland flow velocity (Vfact) is set to 0.013 m/s(=46.8 m/hr) while the river velocity was used in the optimization model to minimize the RMSE.

Because overland flow occurs mostly in the form of small streams, and because subwatershed water loss is estimated using the formula WaterLoss = PFlow*MeanFlowLength*PLossC, which resembles the water loss formula used in river loss estimation, it was decided that the polygon flow loss coefficient PlossC should be set equal to the river flow loss coefficient, LossC.

After these considerations, the river water loss (LossC) and river velocity (Velocity) are selected as the optimization parameters for the simulation model.

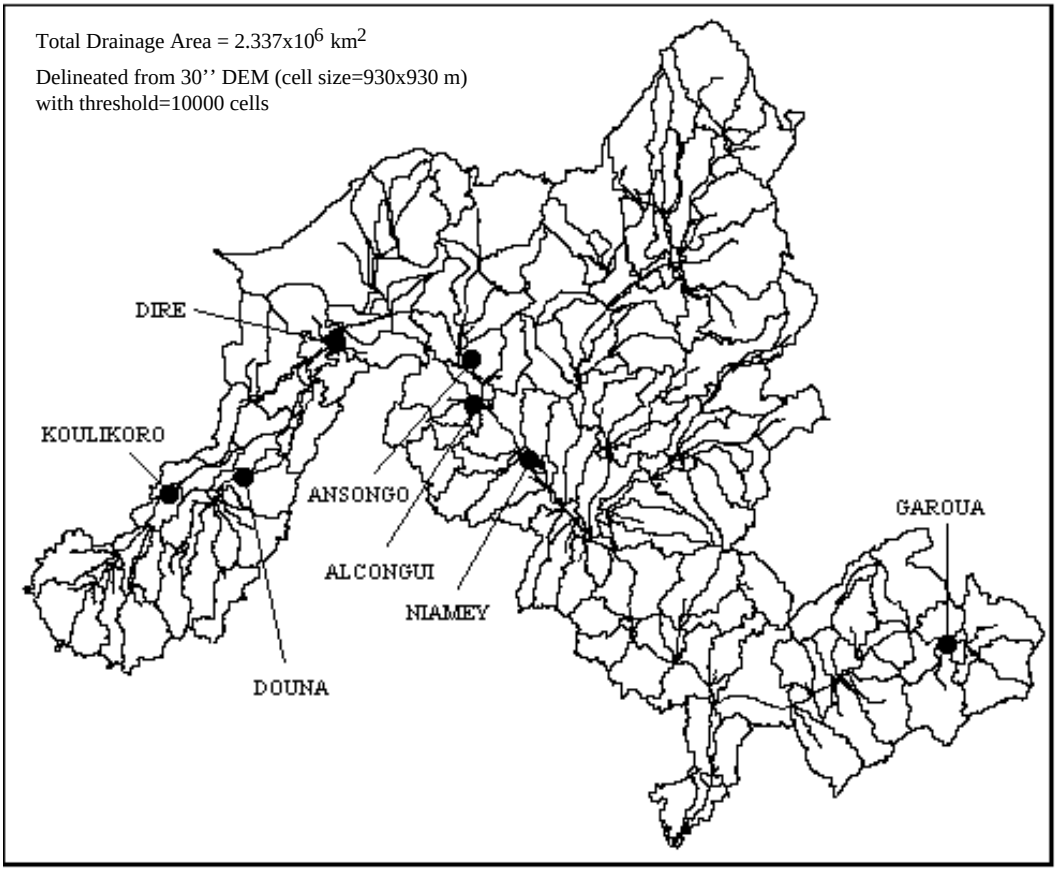


Figure 3.20. The locations of flow-gauging stations in the Niger River Basin

Figure 3.21 shows the converging path of RMSE and SMD of the simulated river flow time-series at Koulikoro when applying the bisection-optimization scheme. Figures 3.22 and 3.23 show the plot of the observed vs. simulated flow time-series when the parameter sets obtained from the optimization model are used. From Table 3.6, it can be seen that when the river velocity is 0.192 m/s (row 11), and Loss-Coefficient for the river and subwatershed is 0.000985 /km, the best fit has RMSE=0.32 and SMD=0.0000116 (row 22). These two values indicate that in a 90 month period, the simulated flow time-series produced the same amount of mass while about 16%(=32%*0.5) of

the mass is incorrectly placed in the time domain when compared with the observed flow time-series.

Figure 3.22a also shows that when using the calibrated parameters, the model underestimates both the high and low flows. The possible explanations of the underestimation may be the following reasons:

(1) in the simulation model, the same LossC value is applied to all the river sections and subwatersheds in the sub-model simulated area,

(2) LossC is kept constant throughout the whole simulation period,

(3) the surplus produced by the soil-water balance model may not be adequate to generate the observed stream flow.

The underestimation caused by reasons (1) and (2) can be corrected to some extent by assigning spatial and temporal variations to LossC but using non-constant LossC will make the optimization model become more complicated. The underestimation caused by reason (3) can be corrected by either using a better soil-water balance model with improved data sets or adopting other means to produce soil-moisture surplus (SurpF(t)) and subwatershed runoff contribution (PFlow(t)). The estimation can probably also be improved by reducing the sizes of subwatershed polygons and the number of subwatersheds selected for each sub-model.

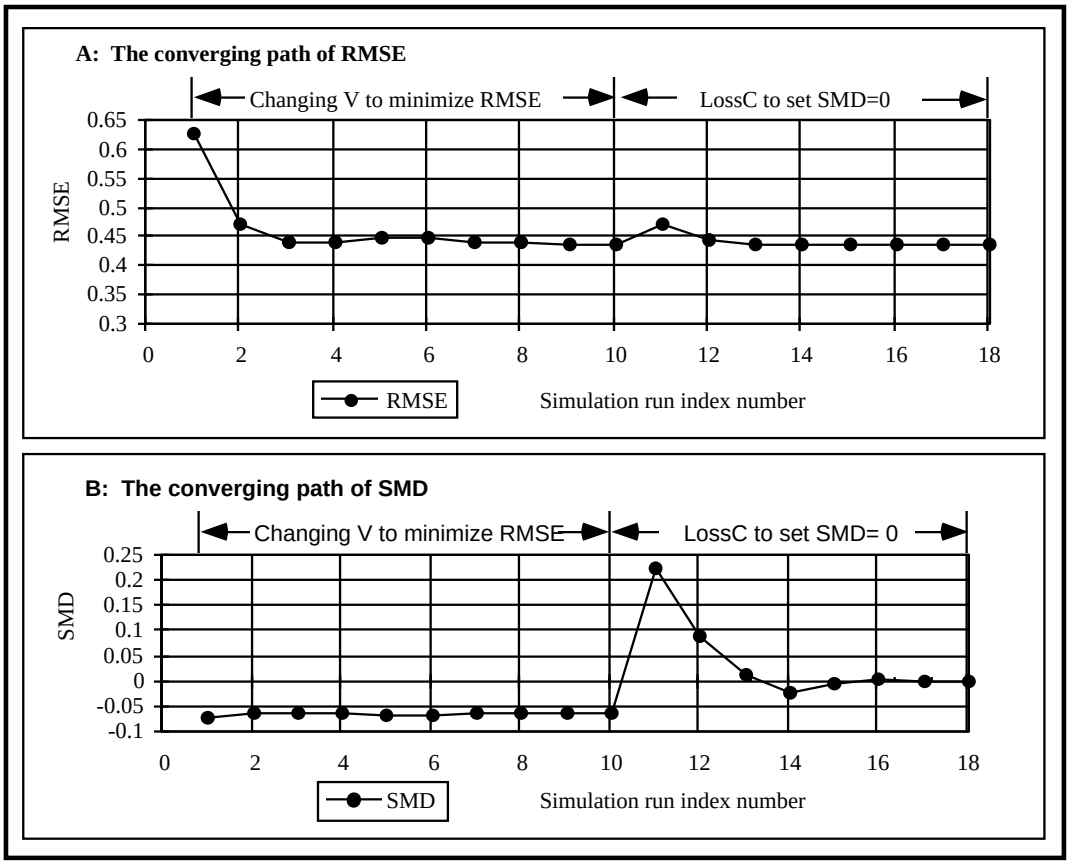


Figure 3.21. Using bisection method to fit two simulation model parameters (LossC and Velocity)

Table 3.6. Optimization of River Flow Velocity and Loss Coefficient

Index	RMSE	SMD	P-value	Parameter-optimized
1	1.1312006	-0.1048283	0.0500000	Ngriver.shp,Velocity,0.05,0.4,1
2	0.4353920	-0.1303611	0.4000000	Ngriver.shpRatio ,Velocity,0.05,0.4,1
3	0.3635078	-0.1295793	0.2250000	Ngriver.shp,Velocity,0.05,0.4,1
4	0.3637302	-0.1295891	0.2260000	Ngriver.shp,Velocity,0.05,0.4,1
5	0.3926416	-0.1277653	0.1375000	Ngriver.shp,Velocity,0.05,0.4,1
6	0.3913918	-0.1278081	0.1385000	Ngriver.shp,Velocity,0.05,0.4,1
7	0.3634192	-0.1289932	0.1812500	Ngriver.shp,Velocity,0.05,0.4,1
8	0.3631668	-0.1290096	0.1822500	Ngriver.shp,Velocity,0.05,0.4,1
9	0.3612605	-0.1293339	0.2031250	Ngriver.shp,Velocity,0.05,0.4,1
10	0.3612916	-0.1293474	0.2041250	Ngriver.shp,Velocity,0.05,0.4,1
11	0.3610867	-0.1291775	0.1921875	Ngriver.shp,Velocity,0.05,0.4,1
12	0.3921203	-0.2030196	0.0007000	Ngriver.shp,LossC,0.0007,0.0013,0
13	0.3200113	0.1924934	0.0013000	Ngriver.shp,LossC,0.0007,0.0013,0
14	0.3198814	0.0100611	0.0010000	Ngriver.shp,LossC,0.0007,0.0013,0
15	0.3484921	-0.0924330	0.0008500	Ngriver.shp,LossC,0.0007,0.0013,0
16	0.3314523	-0.0402009	0.0009250	Ngriver.shp,LossC,0.0007,0.0013,0
17	0.3251588	-0.0148247	0.0009625	Ngriver.shp,LossC,0.0007,0.0013,0
18	0.3224380	-0.0023209	0.0009813	Ngriver.shp,LossC,0.0007,0.0013,0
19	0.3211422	0.0038874	0.0009906	Ngriver.shp,LossC,0.0007,0.0013,0
20	0.3217759	0.0007864	0.0009859	Ngriver.shp,LossC,0.0007,0.0013,0
21	0.3221004	-0.0007651	0.0009836	Ngriver.shp,LossC,0.0007,0.0013,0
22	0.3219352	0.0000116	0.0009848	Ngriver.shp,LossC,0.0007,0.0013,0

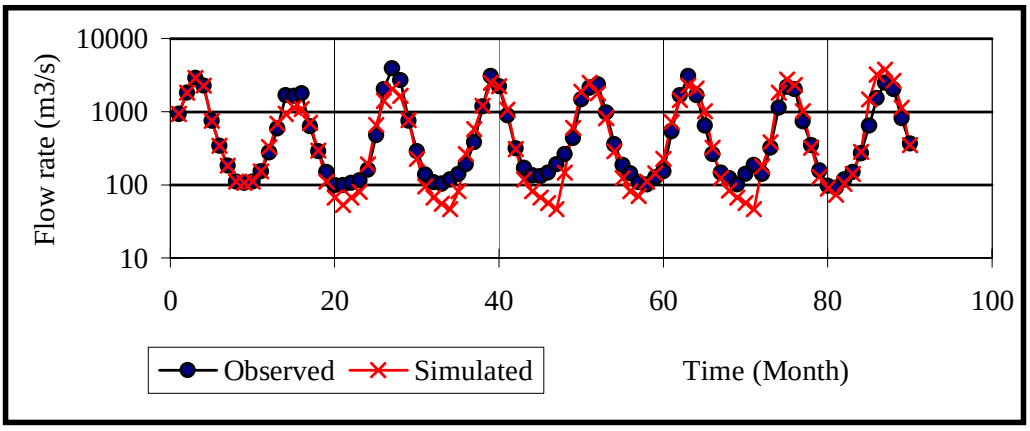


Figure 3.22a. Observed vs. simulated flow time-series at the Koulikoro flow-gauging station (flow rate on base 10 logarithm scale)

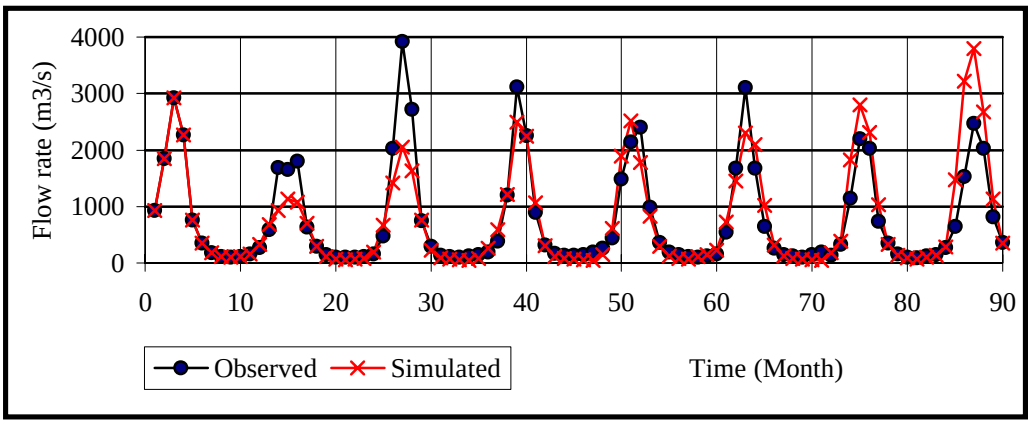


Figure 3.22b. Observed vs. simulated flows at the Koulikoro flow-gauging station

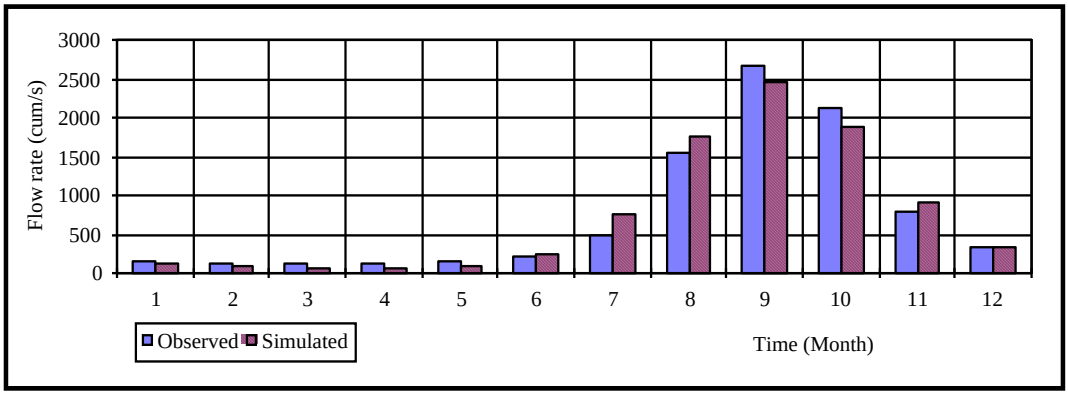


Figure 3.23. Observed vs. simulated mean-monthly flow at the Koulikoro flow-gauging station

Using the same procedure, sub-models were created for the regions associated with four other flow-gauging stations. On each sub-model, the optimization model is used to calibrate two simulation model parameters against the long-term (1961-1990) monthly average flow and flow time-series in a seven and a half year representative period (90 months from July, 1983, to December, 1990). The results of these calibrations are summarized in Table 3.7.

Table 3.7. Calibration Parameter Values for the Sub-Models

Station Name	Time-series	LossC (1/km)	Velocity (m/s)	RMSE	SMD
Koulikoro	90 months	0.00098	0.192	0.3219	0.0000116
	LongTerm	0.001	0.1656	0.1429	-0.0026
Douna	90 months	0.002826	0.0656	0.3736	-0.00394
	LongTerm	0.000964	0.1029	0.1997	-0.0004
Dire	90 months	0.000609	0.184	0.278	-0.00134
	LongTerm	0.00253	0.08	0.3735	0.000167
Ansongo	90 months	0.00027	0.289	0.1986	0.00095
	LongTerm	0.0002935	0.2113	0.093	-0.00287
Niamey	90 months	0.000433	0.4859	0.399	0.00013
	LongTerm	0.00692	0.29	0.262	-0.0008

3.6.4. Flow Interpolation Module

This module is created to interpolate the river flow rates to a flow-check point object. Due to its ability to dynamically segment an arc, the module is also used in the optimization models and dam/reservoir object construction modules to define the location of a point object.

The flow interpolation problem can be described as: given the flow rates at From-Node (FFlow) and To-Node (TFlow) of an arc, find the flow rate at a given point A located on the arc (Figure 3.24). The equation used for the flow rate interpolation at point A can be written as:

$$IFlow_A = FFlow + (TFlow - FFlow) \cdot \frac{SL}{TL} \qquad (3.21)$$

where,

$IFlow_A$ = Interpolated flow rate at point A,

SL = river distance between From-Node and point A,

TL = river distance between From-Node and To-Node, (point a and point g in Figure 3.24).

As shown in Figure 3.24, a river line section consists of a set of straight line segments. Before SL can be computed, it is necessary to identify which segment contains point A. In Figure 3.24, α and β are two angles formed by the line segment and the lines connecting point A to the end nodes of the line segment. It follows that if the line segment contains point A, the condition $((0^o \le \alpha < 90^o)$ and $(0^o \le \beta < 90^o))$ holds. This condition is referred to as "condition-A" in this section. Otherwise, one of the angles is less than 90^o, while

the other angle will be greater than 90^o. It is also true that if the angle formed by two non-zero vectors $\vec{A}$ and $\vec{B}$ is less than 90^o, we have the dot product:

$$\vec{A} \cdot \vec{B} = x_a x_b + y_a y_b > 0 \qquad (3.22)$$

Because in an ARC/INFO coverage, the x,y coordinates of a point are readily available, the dot product and the angle formed by any to vectors can be evaluated using Equation 3.22. Therefore, to compute SL in equation 3.21, the flow interpolation program needs first to find out which river segment contains point A. Using the river section given in Figure 3.24 as an example, the method for computing SL is described below.

Starting from segment ab, the program evaluates the angles ∠Aab, and ∠Aba. Since angle ∠Aba is greater than 90^o, condition-A does not hold indicating that segment ab does not contain point A. Therefore, length ab is added to SL. Applying the same procedure to segment bc reveals that condition-A does not hold for segment bc either so the length of segment bc is also added to SL. On segment cd, the program finds that condition-A holds indicating that the segment contains point A. After adding the remaining segment cs of cd to SL, the program uses Equation 3.21 to linearly interpolate the river flow rate to point A.

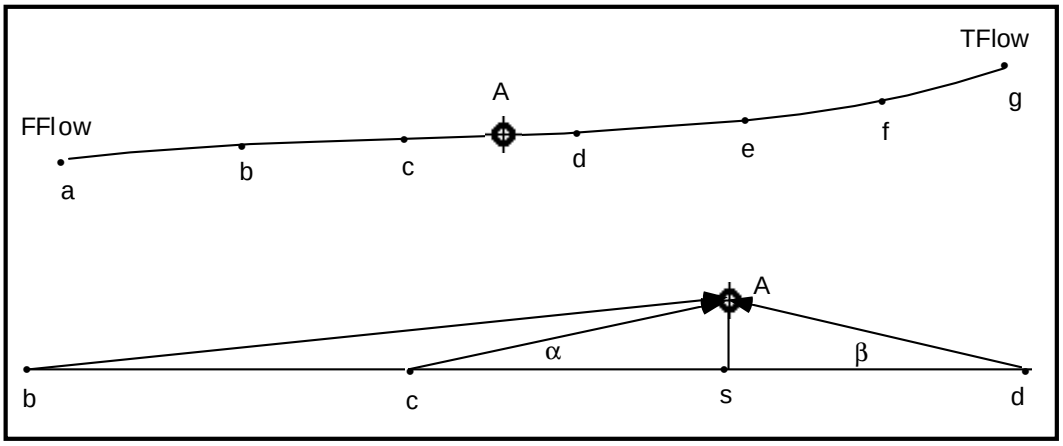


Figure 3.24. Interpolating river flow rate at a given point

The remaining segment cs can be computed using Equation 3.23 (in reference to Figure 3.24):

$$|cs| = |cA| \cdot \cos(\alpha)$$
$$= \frac{(x_A - x_c)(x_d - x_c) + (y_A - y_c)(y_d - y_c)}{\sqrt{(x_d - x_c)^2 + (y_d - y_c)}} \qquad (3.23)$$

3.6.5. Plotting a Longitudinal Flow Profile

The map-based model allows a user to activate tasks directly from a map, greatly simplifying some map-oriented operating procedures. In this and the next section, two map-oriented post-processing modules are introduced to illustrate the strength brought by the integration of programs, maps, and databases.

To perform the hydrologic analysis of a river system, it is often necessary to plot the longitudinal flow profile on a river. If maps and databases are not integrated, the river sections of interest first have to be selected, then the flow data for the selected river sections have to be extracted from the database.

Finally, these data have to be sorted before they can be sent out to plot. The procedures are repeated each time a different river section is picked or flow conditions changed. In addition to the inefficiency brought by this tedious data extraction and data processing procedure, the curve plotted is usually not shown together with the map, which further hinders the interpretation of the plot.

In this map-based simulation model, because a program has access to both the maps and database tables, the longitudinal profile of any selected river sections can be plotted with a simple click on a river section. The logic of the plotting module is described below.

When a river section is selected, its location information is passed on to the plotting program (SFtrplt.pst). Based on the location information and river network connectivity kept by the From-Node and To-Node of each river line, the program traces downstream until the outlet section of the river network is reached. As each river section is found, the flow information related to that river section is collected from the flow table. Because the flow data is collected at the same time river sections are traced, the collected data set is already in the correct order to make the plotting procedure an easy next step. As the plotting program moves from section to section downstream, the section it reaches is highlighted on the map, so that visually, the user can be ensured that no mistake is made in the tracing procedure. The river sections remain highlighted when their longitudinal flow profile plot is on display which helps with profile interpretation (Figure 3.25). When plotting of another river section is desired, the user can simply mark the new selection by clicking at the desired location and let plotting program perform the tasks of river section tracking, data extracting, and curve plotting.

3.6.6. Plotting Time-Series Data at a Selected Location

This program (SFtmplt.pst) is designed to plot the temporal distribution of a physical parameter at a given location. Like the program designed to plot the longitudinal flow profile, this program is also activated directly by a click of mouse on the model maps. When the location information is passed on to the plotting program, the program uses the location information (location ID) to select the time-series (vector) associated with the location and make a plot like that shown in Figure 3.25. The program is designed to work with the spatially-referenced time-series data stored in a database table with the data structure described in Section 3.3.

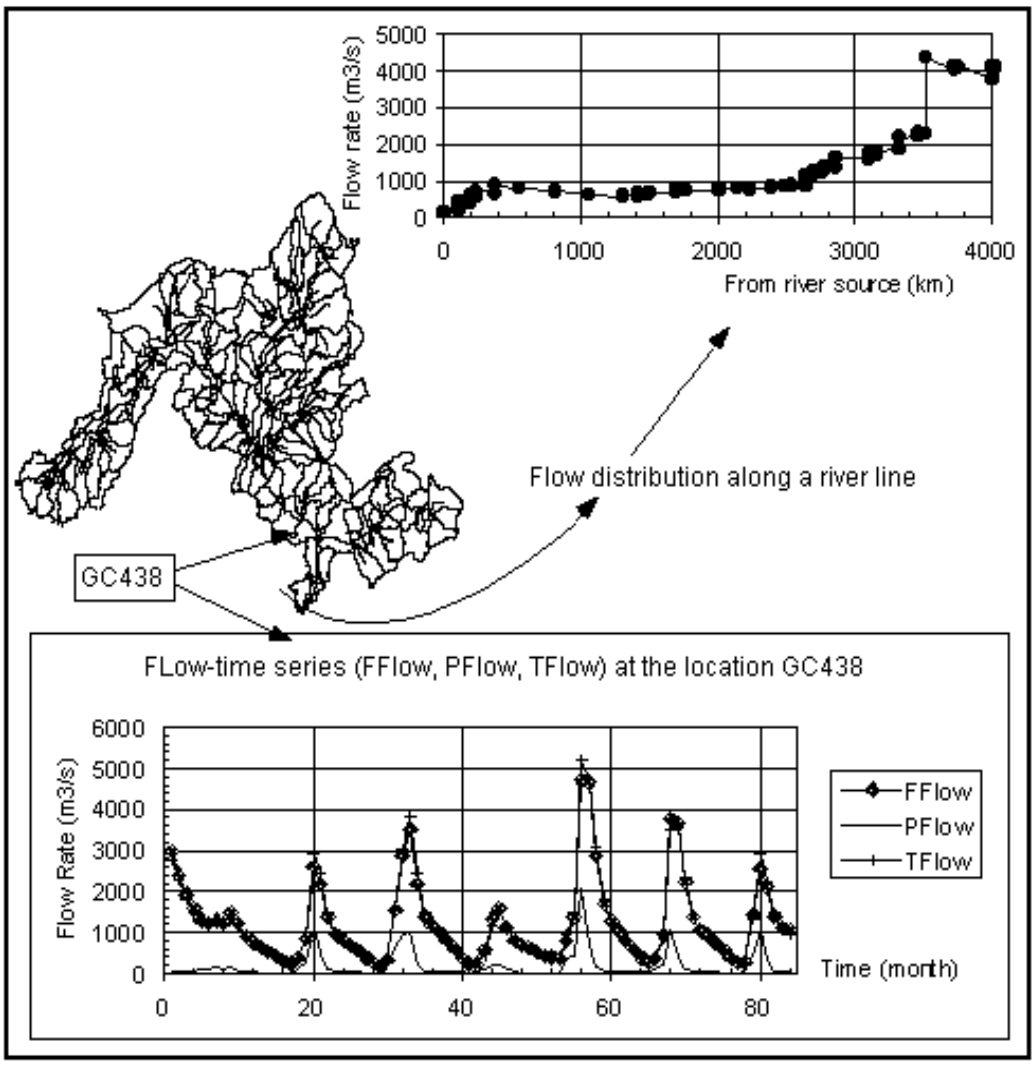


Figure 3.25. Plotting flow distributions with the map-based surface water flow simulation model

3.7. CHAPTER SUMMARY

The map-based surface water flow simulation model is constructed based on the concepts of object-oriented programming and GIS. The model is designed in such a way that all three components of a model, programs, maps, and databases, are integrated.

The basic maps of the map-based model are an arc coverage containing the river lines and a polygon coverage containing the subwatershed polygons. Both maps can be constructed by applying a watershed delineation procedure to a digital elevation model (DEM) of the study region. A sequence of map operations and data structure modifications are performed on these two coverages by a set of pre-processor programs to create river objects, subwatershed objects, and finally, a river network. The connectivity of the river network is maintained by the From-Node and To-Node of each river section on the network. The river sections and subwatershed polygons have a one-to-one relationship.

To simulate river flow, the attributes FFlow and TFlow (FFlow(t) and TFlow(t) for unsteady state) are created for each river object to hold the flow rates at the From-Node and the To-Node of the stream line object. For each subwatershed, PFlow (PFlow(t) for unsteady state) is created to represent the subwatershed's local runoff contribution. The basic relationships between these three quantities are established based on the principle of continuity and are given by Equations 3.1a, 3.1b and 3.2.

The sequential order by which each subwatershed in the stream network is simulated is constructed by a stack-based stream network analysis algorithm developed for this simulation model. The design of this stack-based stream network analysis algorithm is based on the connectivity established by the From-

Node and To-Node of each arc in the network. After the sequential order is decided, Equations 3.3 through 3.12 given in Section 3.4.2. are used to establish the relations between PFlow(t), FFlow(t), and TFlow(t) and simulate water movement on the river network.

Since the stack-based stream network analysis algorithm can be applied to any stream network that fits the assumptions given at the beginning of this chapter, it can be used to simulate other environmental processes as well. For example, if PFlow(t) represents some pollutant sources defined on a polygon object and the pollutant transport mechanism can be established on the rivers, the pollutant mass loading time-series FFlow(t) and TFlow(t) associated with the From-Node and To-Node can be simulated the same way that water flow is simulated. In fact, the process that this model simulates may vary with the type of model used to compute PFlow(t) on each subwatershed.

The integration of programs, maps, and databases allows easy creation of sub-models, making it possible to divide a big study region into several sub-regions for studies with different levels of detail.

To calibrate the simulation model, two optimization models having direct access to the maps, databases, and simulation models are constructed. Of these two models, the interactive optimization model can be used to calibrate no more than 3 parameters at a time because the model needs multiple runs to complete the calibration process, and for each new run the model requires a user who relies largely on the two-dimension plots (Figure 3.18) to define the problem solution space.

The optimization model based on a direction set method is used for all the simulation calibrations in this study. The optimization program is designed in such a way that it does not put a limit on the number of parameters that can be calibrated at one time. But when the number of parameters to be calibrated is

more than four, it may take a long time to complete the optimization procedure. The model also requires a user to provide ranges for all the calibrating parameters at the beginning of a calibration procedure.

Although SMD and RMSE (Equations 3.17 and 3.18) are used in the optimization model to evaluate how well the observed and simulated time-series water, the optimization program is designed in such a way that other standards such as lag-one correlation can be easily added.

The validation procedure of the map-based simulation model is not carried out in this study for the following two reasons:

(1) Because the main purpose of this study is to integrate the three components of a model, the major effort has been devoted to the design and testing of the model programs and the communications between the programs to ensure a smooth integration. A large amount of work is also used to create a smooth pre-processing procedure.

(2) In the Niger River Basin area, it is difficult to find a second set of the required data time-series, such as rainfall distribution and river runoff observations that covers the same length of time without substantial amount of missing records.

The main program (SFlowSim.prc) is designed in such a way that its operation is independent of the modules used to simulate PFlow(t), FFlow(t) and TFlow(t). The model currently provides four modules for simulating water flow on river sections. Because all these four modules are hydrological (lumped) flow routing algorithms, this map-based surface water flow simulation model can be categorized as a hydrological (lumped) flow routing model, comparable to other hydrological (lumped) flow simulation models. If hydraulic (distributed or semi-distributed) flow routing modules such as the methods based on the differential equations of motion and continuity in an open channel (Saint-Venant equations)

were used, this map-based model would become a hydraulic based (distributed) simulation model. At one point of this study, a hydraulic routing module based on the kinematic wave routing method was designed and tested successfully to run with the main program. The module was later disconnected from the main program because not enough field data were available for the Niger River Basin area to support the hydraulic routing module.