Documentation of the Everglades Landscape Model: ELM v2.5

Chapter 10: User's Guide

http://my.sfwmd.gov/elm

July 10, 2006

Chapter 10: User's Guide

10.1 Overview

The ELM is a freely available, "Open Source" project that we hope will be used and modified by others in the scientific community in a collaborative spirit. Other Chapters of the Everglades Landscape Model (ELM) documentation describe the input data, scientific algorithms & source code, model performance, and other material. This Chapter is intended to instruct users on the steps needed to install and apply the ELM in historical (e.g., calibration) simulations.

To use the ELM, one starts with a computer running some "flavor" of the unix operating system (such as Linux). Basic familiarity with unix is required, but advanced expertise is not absolutely necessary. The ELM is installed from a single script that extracts data and code from two compressed file archives. The executable is then built (compiled & linked) from a script, and the model is ready to go.

In the most common/simple application of ELM, a single script is run to verify what output is desired, execute a model run, and archive the results. The user is guided through the several fundamental checks of model output to verify that the model indeed performed as expected. The outputs are described, covering a range of spatial and temporal scales of the landscape. Their interpretation is dependent on an understanding of the science of ELM covered in the other Chapters of this documentation.

As should be apparent from this and other Chapters of the model documentation, the ELM was designed to be applied by modifying databases, not the model source code. "User-friendly" supporting databases are available to select different outputs, change parameters, or explore/edit aspects of the supporting data. However, those databases need not be immediately opened/modified, depending on the user's initial interest.

A few of the more advanced applications of ELM are covered in brief, but are generally beyond the scope of this User's Guide. These topics include the automated sensitivity analysis of the model, the creation of new subregional applications, and evaluating scenarios of future restoration alternatives. While these applications of the model are all data-driven and relatively straightforward, the details of changes to data and requisite quality assurance are left to a subsequent extension of this guide.

10.2 Computing environment

The ELM is truly a multi-platform simulation model, capable of running in a variety of computing environments without modification. No changes to the C source code, scripts, or "makefile" are needed to move among any of the computing environments that we have tested. The compilation and run scripts detect the type of unix operating system, with no user intervention. This allows the ELM developers to modify one set of code (stored on one file system), and routinely compile & run the OS-specific executables from any available platform. The production environment for ELM is Red Hat Linux on an Intel chip, while Apple OS X (Darwin) is a very useful modeling environment.

10.2.1 Hardware

The ELM can be installed and executed on any one of a variety of common hardware architectures that have some form of unix[1] available (Table 10.1 below). Available storage on the file system (hard disk) should be at least 600 MB: roughly 500 MB is needed for all of the input data/databases and source code, while a 20-year run with basic outputs, including animated monthly time series of a handful of variables, uses approximately 100 MB disk space. Different subregional applications (of various grid sizes) vary the memory (RAM) requirements, but the regional ELM application that is in the standard distribution uses less than 90 MB RAM, irrespective of the simulation length.

10.2.2 Software

No commercial software is necessary. The only requirement to install and use ELM is a unix operating system that includes a gcc[2] compiler. No custom libraries need to be modified/installed beyond those already available in a standard operating system installation with a functional compiler. Tools that are technically "optional", but highly desirable, include a Geographic Information System (the Open Source GRASS GIS is recommended), and spreadsheet software (Open Office Calc is recommended). For optional/recommended software tools, see Appendix: Software recommendations.

Table 10.1. ELM compilation and execution has been tested in these environments. At the unix command line, type "gcc --ver" and "uname -a" for this information.

Compiler	Operating System	OS release version	CPU
~~gcc v.3.2~~ (unsupported)	(unsupported) ~~Sun Solaris~~	5.8	sparc
gcc v.3.2.2	Red Hat Linux	2.4.20-27.9smp	i686 (Pentium)
gcc v.3.3	Apple Darwin	6.8	Power Mac (G4)
gcc v.3.3.3	SuSE Linux	2.6.4-52-default	i686 (Celeron)
gcc v.3.4.3	Red Hat Linux	2.6.9-5.ELsmp	i686 (Pentium)

[1] Our available Sun Solaris and Apple Darwin platforms are outdated, and thus we have not tested the ELM code in more recent versions of these OSs & associated standard libraries.

[2] GNU Compiler Collection, gcc, at http://gcc.gnu.org/ There are no compiler-specific dependencies, and thus other ANSI C compliant compilers should be compatible with ELM code.

10.2.3 Runtimes

One of the platforms available to the ELM developers is an inexpensive Dell™ laptop with an Intel Pentium™ 2.66 GHz processor. On this computer, the run-time for the regional-ELM implementation (10,394 grid cells @1 km^2 resolution), with standard output, is slightly over 1 hour for a 20-year simulation.

10.3 Installing the model

Using an Open Source[3] philosophy, we hope to encourage collaboration in the modeling community. Towards that end, the source code and data are available for download on the ELM web site, and all C source code in the ELM project is documented in detail using the automated "Doxygen" web-based documentation system (see Model Structure Chapter).

10.3.1 Standard

The ELM project is installed in a directory of the user's choosing, without affecting existing operating system "libraries" or other components of the user's file system. To install the ELM, one places the code & data archives into an empty directory, and runs a single script, by following these steps (replacing "X.Y" with "2.5" for ELM v2.5):

1) Obtain the code and data (from CD or http://my.sfwmd.gov/elm)
    a. ELMinstall.sh (installer shell script)
    b. ELMX.Y.data.updateA.B.tar.gz (compressed archive of data, ELM version X.Y, update A.B)
    c. ELMX.Y.src.updateA.B.tar.gz (compressed archive of code, ELM version X.Y, update A.B)
2) Make a home and install your project
    a. Create an empty directory anywhere on your file system, put above 3 files into it, and "cd" into that directory
    b. Run the install script on unix command line: "./ELMinstall.sh"
    c. Note: the install script guides you on how to set up the several environment variables that are needed. One is "$ELM_HOME", which is the absolute path of the directory in which you placed the project.
3) Build the executable
    a. Run the build script on unix command line (ELM version X.Y): "./build ELMX.Y"

10.3.2 Custom

The standard installation is generally all that is needed. However, the user has more flexibility in choosing the location(s) of model output, along with customization of other characteristics of the model. Note that the choice of operating system does not influence any of the installation procedures. For the details of the potential customizations, see this

[3] http://www.opensource.org/

Chapter's Appendix: Environment variables and Directory/file structure.

10.4 Running the model

The ELM is run from the unix command line through the use of "shell" scripts. Basic familiarity with unix is required, but advanced expertise is not absolutely necessary.

10.4.1 Quick start

For those who want to run a simulation "right now" using the defaults set in the standard distribution of source code and data, simply jump in and invoke a script (after installing the model as described above!). In the commands below, replace "X.Y" with "2.5" for ELM v2.5. For all commands and filenames, remember that unix is case-sensitive.

1) Invoke the Run script, responding to its prompts (ELM version X.Y): "./Run ELMX.Y myFirstRun"
2) The Run script asks you a couple of questions. Say no to both for now: the model will run, and then the results will be archived in a new directory called "myFirstRun", within the archive directory "$ELM_HOME/arc_out/"
3) Check/interpret the output as outlined in the "Output" section later in this Chapter.

10.4.2 Runtime configuration files

There are two model configuration (text) files[4] that can be modified prior to running the model. One file, "Driver.parm", is edited to select which ecological module(s) to execute, set the starting and ending dates of simulation, and other such model settings. The other, "Model.outList", is edited to select which variables to output, their output type & location, and output frequency. These two configuration files are directly read by the model during the initialization sequence.

10.4.2.1 Driver.parm

This is the primary configuration file, providing significant flexibility to the user. Some of the more common changes that may be made in this configuration are:

- change location of model output
- change start and end dates of simulation
- change output intervals for budgets, canals, and internal variable averaging
- turn on/off habitat switching module
- turn on/off water management modules
- turn on/off various hydro-ecological vertical solution modules
- run sensitivity analyses on parameters

This text file is self-documented at a brief level of detail. This Chapter's "Appendix:

[4] in $ELM_HOME/SME/Projects/ELMX.Y/RunParms/

Driver.parm" expands on the information for each of these runtime parameters.

The "Check" script is used to quickly check these settings, and edit them if desired (using the standard unix text editor "vi").

10.4.2.2 Model.outList

This text file is exported from the "ModelOutlist_creator_*version*.xls" interface. That spreadsheet database is found in the "$ELM_HOME/SME/Projects/Dbases/" directory. It is "user-friendly" and fully self-documenting, and is perhaps most commonly used for initially selecting and configuring the different output command options. For basically any dynamic variable in the model, the user can select the following[5] combinations of commands to produce output:

- map time series (animations): "G( )" command
- scale the values of map time series output: "S( )" command (required w/ "G( )")
- point time series (individual grid cells): "P( )" command
- time interval for output (independent for each variable): "O( )" command

The map time series consists of multiple domain-wide spatial maps of the selected variable at the selected output interval, with each variable's multi-file time series put in a separate output directory ("./Output/*.*/"). The point time series are put in the "./Output/PtSer/" output directory, with a time series at the selected output interval in a separate file for each variable, with each file containing multiple points (grid cells).

Note: summaries of all canals & water control structures ("./Output/Canal/"), and all user-defined Basin/Indicator-Region data ("./Output/Budget/") are always output. The user can modify the output frequency of those data via the "Driver.parm" configuration. (Basins and Indicator Regions are defined in an input map; see the "Modifying Data" section of this Chapter).

Although it is relatively quick and easy to use, it is not necessary to routinely use the ModelOutlist_creator spreadsheet interface: once a user becomes familiar with the output commands, the "Check" script can be used to most quickly check the settings in the "Model.outList" text file, and edit them if desired (using the unix text editor "vi").

10.4.3 Scripts

The following are the scripts that are available for a variety of tasks associated with using the ELM. Most of the scripts are "stand-alone", but are designed in a modular fashion so that they can also be controlled by higher-level calling scripts. (For example, the "Run" script shown above is a main controller script that calls the stand-alone scripts of "Check", "go", and "ArchiveRun", while those latter scripts call others such as "PathModel"). Table 10.2 describes the script usage and hierarchy.

[5] The map time series that are produces are in "unsigned character" binary formats that produce the smallest file sizes, and the output maps are scaled by the user via the interface. Hierarchical Data Format (HDF) was supported in earlier versions of ELM, but is not updated for ELM v2.5. Subsequent versions will support either "hdf" or "cdf" formats.

Table 10.2. Scripts used in the ELM project. The three grey-shaded scripts are all that are needed to install and run the ELM. Scripts are modular and nested in a hierarchy; all scripts can be executed as stand-alone applications (w/ 1 exception). Syntax: *ProjName* is the name of the ELM project (e.g., ELM2.5); *runName* is a user-defined name to denote a particular simulation run

	Primary script	Secondary script	Syntax	Included/called scripts	Purpose of script
Model installation					
	ELMinstallX.Y.sh		ELMinstall*X.Y* .sh, where *X.Y* is model version	none	Install the ELM project in the user's directory. Fully self-documented. (script name came w/ distribution)
	build		build *ProjName*	PathELM_HOME, PathModel, PathOSTYPE	Builds an executable of the model project from the make file (compiles, links source code).
Model run					
	Run		Run *ProjName runName*	Check, go, CopyInput, ArchiveRun, PathELM_HOME, PathModel, PathOutput, PathArchive	Controller script that configures, runs, and archives a simulation.
		Check	Check *ProjName*	PathELM_HOME, PathModel	View and change the model runtime configuration.
		go	go *ProjName*	PathELM_HOME, PathModel, PathOutput, PathOSTYPE	Simply runs the model executable. NOTE: output from a simulation run made via this script is OVERWRITTEN by a subsequent invocation of this script; use ArchiveRun script to save a simulation.
		ArchiveRun	ArchiveRun *ProjName runName*	PathELM_HOME, PathModel, PathOutput, PathArchive, mkOutDirs	Archives a simulation's output and input as a "keeper" under a user-defined name. It moves all output files and copies selected input files to a user-defined new directory in the $ELM_ARCHIVE_PATH.
	finishOutList		finishOutList *ProjName target* , where *target* is file made by ModelOutlist_creator.	PathELM_HOME, PathModel	If ModelOutlist_creator was used: Does the final processing needed on the Model.outList text file that was created by the ModelOutlist_creator workbook (OpenOffice/Excel).
Model distribution/backup					
	ArchiveData		ArchiveData *ProjName descript* , where *descript* is descriptive identifier	none	Archives all input data required for ELM historical (e.g., calibration) runs to a compressed tar archive. Used for ELM-version distributions. To use, modify source and target directories in the script.
	ArchiveSrc		ArchiveSrc *ProjName descript* where *descript* is descriptive identifier	PathELM_HOME	Archives all required ELM source code to tar archives in two locations: an uncompressed one in $ELM_HOME, and a compressed one in a remote directory. Used for ELM-version distributions. To use, modify destination directory in the script.
Utility					
		PathArchive	PathArchive	none	Checks validity of $ELM_ARCHIVE_PATH for model archiving and exports it if needed.
		PathELM_HOME	PathELM_HOME	none	Determines if the (fundamental) $ELM_HOME variable appears valid.
		PathModel	PathModel	PathELM_HOME	Checks validity of the base $ModelPath for the model Project data/executable, and creates & exports that path if needed.
		PathOutput	PathOutput *ProjName*	PathELM_HOME, PathModel	Checks for validity of an existing OutputPath for model output (defined in Driver.parm file) and exports it if valid.
		PathOSTYPE	PathOSTYPE	none	Determines if the $OSTYPE variable reflects a tested platform. (The name of the script is for consistency with similar script names, and OSTYPE is only used in relation to paths/filenames).
		CopyInput	NA (is not stand-alone)	none	Only called from the "Run" script. It has 2 primary purposes: 1) Force the user to write some Notes on simulation about to be run; 2) Create named copies of frequently-changed data files.
		mkOutDirs	mkOutDirs *OutputPath ProjName*	none	Create the required output directory names if they have been removed from the model Project OutputPath.
	rmAnim		rmAnim *OutputPath ProjName*	none	Delete all files in Animation* directories for a project in a given path. For a measure of safety, this is only used as a stand-alone script, and the user needs to manually type in the path, then confirm the deletions.
Advanced: acquire water control structure flows					
	getDSSflow		getDSSflow	none	Acquire flow data. Full instructions for advanced applications not in this current documentation.
	StrNames		StrNames	none	(Compiled binary) to extract names of structures from a "DSS" catalog. Full instructions for advanced applications not in this current documentation.
Advanced: GRASS for animations, vector canal input/visualization, other					
	AnimGrass		NA (not distributed, FYI only)	PathOutput, PathArchive	GRASS (script not distributed): Links model output to a GRASS directory in preparation for animation using xganim.
	AnimGrassNow		NA (not distributed, FYI only)	none	GRASS (script not distributed): Runs xganim within GRASS.
	AnimGrass_rm		NA (not distributed, FYI only)	none	GRASS (script not distributed): Deletes the links to model output and the other GRASS animation files for a particular variable.
	reachin		NA (not distributed, FYI only)	none (but uses ELM variable "$ModelPath")	GRASS (script not distributed): Creates GRASS ascii vector files for all canals contained in the CanalData.chan ELM-input file.
	reachinvect		NA (not distributed, FYI only)	none	GRASS (script not distributed): Import ALL reaches from ascii into Grass binary vector format.
	reach_calib_v2.4		NA (not distributed, FYI only)	none	GRASS (script not distributed): Display canal reaches in distinguishing colors, and show thel water control structures.

10.5 Input data modification

Several databases are used to modify and document a variety of important components of the ELM. The purpose of this section is to call the user's attention to these self-documenting databases, which are *critical to the use of the ELM, particularly when learning the model*. Other data sources (described in the Data Chapter) are used for time series data that are input to the model, and some form of GIS (below) is needed to visualize and modify the spatial maps that are input to the model.

10.5.1 Databases

Our goal has been to create a system of integrated, relational databases using the Open Source MySQL. However, these prototype databases are not ready for release, and we instead use the Open Office Calc spreadsheet software[6] to perform the necessary data management functions. Table 10.3 provides an overview of the primary functions of these data management systems.

Table 10.3. Spreadsheet-based databases used in a) data maintenance and documentation of model parameters and model variables, b) generating source code of model, and c) generating output configurations for model runs. Databases are found in $ELM_HOME/SME/Projects/Dbases/.

Database name	**Database functions**
GlobalParms_vX.Y.xls	1) Maintain and document (incl. units and source/metadata) parameters that are globally distributed across model domain. 2) Generate code of header file, transferring parameter documentation to model source code. 3) Generate upper and lower values of all parameters for automated sensitivity analysis.
HabParms_vX.Y.xls	1) Maintain and document (incl. units and source/metadata) parameters that are specific to different habitats in the model domain. 2) Generate code of header file, transferring parameter documentation to model source code. 3) Generate upper and lower values of all parameters for automated sensitivity analysis.
ModelOutlist_creator_vX.Y.xls	1) Generate all input-configuration commands for any model variable, map and point time series output. 2) For all Everglades monitoring sites, calculate model grid cell row-column (at any grid scale) from its geographic coordinates. 3) Maintain and document (incl. units and source/metadata) all variables used in the model. 4) Generate code of multiple header files, transferring documentation of variables to model source code.

[6] fully compatible with Microsoft Excel

The single exception to the use of Open Source software is our database ($ELM_HOME/SME/Projects/Dbases/Structs_attr_vX.Y.fmp) of the attributes of water control structures, for which we continue to use FileMaker Pro software. This database continues to be very useful in creating new subregional applications or modifying water control structures for evaluating alternative water management scenarios. However, it is not essential to the use of ELM in the mode intended for this User's Guide Chapter: the water control structure attributes for the current simulations are documented through snapshots of the records for all of the necessary water control structures, and the text input file can be viewed or modified using spreadsheet software (see Data Chapter).

10.5.2 GIS

Any software capable of reading raw/generic binary data arrays can be used to edit/visualize the map inputs. The ELM developers use the GRASS GIS (see Appendix: Software recommendations). Through the use of unix symbolic links between the GRASS and the ELM data directories, the ELM directly reads GRASS project data files (uncompressed binary data and text header) as model input. However, no GRASS-specific encoding of binary information is used, and thus the data files may be opened with any program that can read raw binary data arrays. Scripts are available to directly input and visualize the ELM (text) canal vectors in GRASS.

There are three sub-directories within an ELM project's input ./Data/ directory: Map_bin, Map_head, and Map_hist. The model reads each raw binary data file in the Map_bin subdirectory, and reads its associated header description in the Map_head subdirectory. The history and other pertinent metadata are in the Map_hist subdirectory, but that information is not used in the model.

All spatial data are referenced to zone 17 of the Universal Transverse Mercator (UTM) geographic coordinate system, relative to the 1927 North American Datum (NAD). The ELM regional application uses 1 km^2 square grid cells that encompass an area of 10,394 km^2 (4,013 mi^2) in the active domain. All of the maps of the regional application are bounded by a rectangle of UTM coordinates in zone 17 (NAD 1927), as shown in the lines in the below regional-domain example of the text header files:

zone:	17 (UTM zone)
northing:	2,953,489 m (UTM north coord)
southing:	2,769,489 m (UTM south coord)
easting:	580,711 m (UTM east coord)
westing:	472,711 m (UTM west coord)
columns:	108 (number of columns in 2D array)
rows:	184 (number of rows in 2D array)
east-west resol.:	1000 m (grid cell length in 2D array)
north-south resol.:	1000 m (grid cell width in 2D array)
format:	*"X"* bytes/cell, as defined below
compressed:	0 (no compression)

The *"X"* value of "format" of the raw binary data is one of the following:

0.:	1 byte per grid cell
1:	2 bytes per grid cell
3:	4 bytes per grid cell

10.6 Output

During the initialization phase of a simulation, the various configurations that the user chose are echoed to the console (screen). Subsequently, the simulation date is iterated on the console as the computations are made. A successful simulation will end with the following message printed to the screen:
"END. The simulation(s) took zz.zzz minutes to run using your yyyyy OS box.", followed by other messages depending on the scripts that are running.

10.6.1 Quick start

Upon completion of (or during) a simulation, the user is advised to make the following minimal checks to verify that the simulation was "well-behaved".

4) To verify that no errors were in the simulation, search the "Debug/Driver1.out" text file (see below) for the all-caps string "ERROR", which can be the full word or part of a word (i.e., "capacityERROR");
5) View the "Budget/budg_Wcm1" text file, and verify that the cumulative mass balance error variables, "SumERR_*" for each Basin & Indicator Region identity, is reasonable, i.e., on the order of tens of microns height.
6) Peruse one of the spatial time series of map outputs to verify the spatio-temporal dynamics "pass the laugh test" . Viewing an animation, or individual maps, of the "SfWatAvg" (surface water depth, averaged during output intervals) variable is a good choice - assuming the user kept that variable's output commands in the Model.outList configuration.
7) Dive into the other output files as desired, using the below descriptions as your guide.

10.6.2 Output file structure

During a simulation, all output is always written to the "Output/" directory in the user's output path (Table 10.4). After a simulation terminates, the output may be moved (archived) to the user's archive path via the "ArchiveRun" script (which is also called by the "Run" script). In the below directory descriptions, "ProjName" is the Project name (such as ELM2.5) that was input by the user on the command line.

Un-archived output location: "OutputPath/ProjName/Output/", where "OutputPath" is the absolute path to model output, changeable in the "Driver.parm" file.

Archived output location: "$ELM_ARCHIVE_PATH/ProjName/runName", where "$ELM_ARCHIVE_PATH" is the archive path set up by the user[7], and "runName" is a user-defined name to denote a particular simulation run.

[7] "$ELM_ARCHIVE_PATH" was set up by the user when installing the ELM. The location may be set to anywhere, but initial installation was in "$ELM_HOME/arc_out/"

Table 10.4. Output directories and description of files they contain. These directories are relative to the un-archived or the archived output locations described above.

Output directory	Output description
Animation1...Animation60	Map time series for individual variables, with separate directory for each variable. After archiving a simulation, non-empty directories are moved & renamed with the variable names.
or: VariableA, VariableB, ...	Map time series for individual variables, with separate directory for each variable. Prior to archiving a simulation, the directory names are simply Animation1, Animation2, ..., Animation60 (maximum).
Budget	[BIR] Time series of budgets and pre-set Performance Measures in Basins/Indicator Regions (BIR).
Canal	Time series of a) canal depths and constituent concentrations, and b) water control structure flows and constituent concentrations.
Debug	Variety of detailed output for debugging and error checking.
PtSer	Time series of individual variables at point (grid cell) locations distributed through model domain.

10.6.3 Debug (errors and warnings)

The "Debug/" directory will always contain at least two debug-related files. Truly critical errors (such as missing inputs, memory constraints, etc) will terminate the simulation with an informative message. Numerical errors or warnings do not necessarily terminate the simulation (in order to allow the user to debug the problem). It is important to monitor the Driver*.out files for any errors or warnings, particularly after configuring a new application:

- Driver0.out: text file that echoes input data that were successfully read, including simulation start-end dates, hydro-ecological parameters, output configurations and others.
- Driver1.out: text file that contains a variety of warnings, error messages; details of model output are printed, depending on the level of the "debug" parameter (in "Driver.parm", see Runtime configuration section of this Chapter).

The "Debug/" directory will contain two debug-related files when running the Water Management modules:

- ON_MAP_CANAL.txt: a tab-delimited 2D array text file of the modifications to the "ON_MAP" file that was done by the "Canal-marsh flux module of the Water Management code (see Model Structure Chapter).
- CanalCells_interaction.txt: text file of list of cells that interact with each canal reach

10.6.3.1 Postprocessing Debug text files

All files in the “Debug/” directory are text files. The Driver*.out files are intended to be searched/queried using any text editor. The “ON_MAP_CANAL.txt” file is best visualized after import into any spatial mapping program or GIS (such as GRASS).

10.6.4 Spatial: Basin & Indicator Region (BIR) time series

Budgets and preset Performance Measure variables are output at the different spatial scales defined by the hydrologic Basins and Indicator Regions (BIR) input map. As discussed in the Model domains section of the Data Chapter (“basins” input Data file), hydrologic basins are “parent” regions that (may) contain “child” Indicator Regions, and parent basins’ data include (e.g., sum) the data on all child Indicator Regions contained within them. Basin 0 is the entire model domain. Well-drawn BIR spatial distributions are particularly useful for evaluating output dynamics (budgets and Performance Measures) along ecological gradients. Table 10.5 provides an overview of the budget and Performance Measure variables in each of the output files.

Table 10.5. Budget and preset Performance Measure variables in Basins & Indicator Regions (BIR). The variables are output for each BIR in the individual files.

Budget file name	**Budget file description**	**Budget variables for each BIR**																						
budg_Wacr1...5	Water budget: all storages, acre-feet units	total volume - prior interval	total volume - current interval	rain-in	evaporation-out	transpiration-out							structure-in	structure-out	overland-in	overland-out	levee seepage-in	levee seepage-out	ground water-in	ground water-out	error-current interval	error-cumulative sum	avg-inputs	avg-outputs
budg_Wcm1...5	Water budget: all storages, height (cm) units	total volume - prior interval	total volume - current interval	rain-in	evaporation-out	transpiration-out							structure-in	structure-out	overland-in	overland-out	levee seepage-in	levee seepage-out	ground water-in	groundwater-out	error-current interval	error-cumulative sum	avg-inputs	avg-outputs
budg_S1...5	Salt/tracer budget: all storages, mass-per-area units	total mass - prior interval	total mass - current interval										structure-in	structure-out	overland-in	overland-out	levee seepage-in	levee seepage-out	ground water-in	groundwater-out	error-current interval	error-cumulative sum	avg-inputs	avg-outputs
budg_Par1...5	Phosphorus budget: all storages, mass-per-area units	total mass - prior interval	total mass - current interval	rain-in									structure-in	structure-out	overland-in	overland-out	levee seepage-in	levee seepage-out	ground water-in	groundwater-out	error-current interval	error-cumulative sum	avg-inputs	avg-outputs
budg_P1...5	Phosphorus budget: all storages, mass units	total mass - prior interval	total mass - current interval	rain-in									structure-in	structure-out	overland-in	overland-out	levee seepage-in	levee seepage-out	ground water-in	groundwater-out	error-current interval	error-cumulative sum	avg-inputs	avg-outputs
budg_Pwat1...5	Phosphorus budget: water-borne storages, mass-per-area units	total mass - prior interval	total mass - current interval	rain-in	settle-out	surfwat. mineralization-out	porewat. mineralization-out	periph. uptake-out	macroph. uptake-out	desorb-in	sorb_to-out										error-current interval	error-cumulative sum	avg-inputs	avg-outputs
budg_Pliv1...5	Phosphorus budget: live biotic storages, mass-per-area units	total live mass -prior interval	total live mass current interval	macroph. mass-current interval	calc.periph. mass-current interval	noncalc.periph. mass-current	macroph. NPProd.-in	calc. periph. GPProd.-in	noncalc. periph. GPProd.-in	macroph. mortality-out	calc. periph. mortality-out	noncalc. periph. mortality-out									error-current interval	error-cumulative sum	avg-inputs	avg-outputs
budg_Pded1...5	Phosphorus budget: dead biotic storages, mass-per-area units	total dead mass -prior interval	total dead mass -current interval	macroph. ToSoil-in	periph. ToSoil-in	decomp.-out	settle-in			sorb_to-in	desorb-out										error-current interval	error-cumulative sum	avg-inputs	avg-outputs
Perf. Measure file name	**File description**	**Performance Measure variables for each BIR**																						
BIRavg1...5	Hydro-ecological Performance Measures	surface water depth	unsaturated water depth	TP conc. surface water	TP conc. pore water	TP conc. soil	noncalc. periph. biomass	calc. periph. biomass	macroph. biomass	land elevation														

10.6.4.1 Budgets (in BIR)

The "Budget/" directory contains tab-delimited text files with budgets of water, phosphorus, and salt/tracer in the BIRs. The reporting time interval is selected by the user (see Runtime configuration section of this Chapter). In each budget, all inflows and outflows to/from each BIR are summed for the relevant variables within each reporting interval. For example, a 30-day reporting interval will result in a hydrologic budget that reports the sum of the different inflows (rain, seepage inflow, etc) and outflows (ET, seepage outflow, etc.) within each 30-day period during the simulation. Numerical errors in mass conservation[8] are always calculated for all budgets, both cumulative during each reporting interval, and cumulative across the model simulation period.

10.6.4.2 Preset Performance Measures (in BIR)

The "Budget/" directory also contains tab-delimited text files with preset Performance Measure averages in BIRs. The reporting time interval is selected by the user (see Runtime configuration section of this Chapter), and is used to calculate the daily arithmetic mean value of each performance measure within the interval. These Performance Measures include hydrologic, biogeochemical, and biological dynamics within the region.

10.6.4.3 Postprocessing BIR text files

All BIR budget and Performance Measure files are in tab-delimited text format, and thus can be directly read into any spreadsheet program such as Open Office Calc or Microsoft Excel. The primary method for ELM postprocessing is the use of scripts written in the Python scripting language. The ELM developers have a flexible set of Python postprocessing scripts that will produce a variety of summaries of these data for visualization and analysis, but that development is not complete enough for release. Spreadsheet templates for different summaries of the output data are available from the developers, but are unsupported.

10.6.5 Spatial: Domain-wide map time series

Virtually any variable in the model may be output as domain-wide maps at a user-specified output interval (see Runtime configuration section of this Chapter). These maps may then be analyzed individually, summarized across time, or animated using visualization software.

If a simulation has not yet been archived, the output maps of any user-selected model variable are placed in one of the AnimationZZ directories in the Output directory, where "ZZ" is an integer between 1 – 60. As described earlier, the model archiving process

[8] The *maximum* magnitude of cumulative errors in mass balance of water storage dynamics ranges within the order of (positive or negative) 1 to 10 microns, depending on the cumulative interval (monthly or multi-decade period-of-simulation), the presence/absence of canal interactions, and the spatial scale of the budgeted region. The *maximum* magnitude of cumulative errors in mass balance of phosphorus storage dynamics ranges within the order of (positive or negative) 0.001 to 0.01 ug/m^2, depending on the cumulative interval (monthly or multi-decade period-of-simulation), the presence/absence of canal interactions, and the spatial scale of the budgeted region.

renames the directories to those of the variable it contains.

10.6.5.1 Postprocessing map files

As configured by the user via the ModelOutlist_creator interface (see Runtime configuration section of this Chapter), all output maps are 2D rectangular arrays in generic/raw binary format (i.e., they are not encoded with any software-specific attributes).

To save significant disk space compared to floating point arrays, the map files are output as "1-byte, unsigned integer" data. In any given directory containing a time series of maps of a given variable, the numeric values in the 2D arrays range from 0-255. The value of "255" is reserved for grid cells that are "off-map", or outside of the active domain. The parameters in the scaling equation chosen by the user (via the ModelOutlist_creator) for each output variable must be used to rescale the integer maps back to the actual (floating point numbers and) units of the model using the equation:

model_floatValue = outMap_intValue * Multiplier + Offset,

where model_floatValue is the actual value of the floating point number calculated in the model, outMap_intValue is the integer number stored in the map array, and Multiplier and Offset are the scaling multiplier and offset, respectively, input by the user in the Model.outList. The units of the model_floatValue for each variable were given in the ModelOutlist_creator interface. For example, ponded surface water depth (SURFACE_WAT) is often scaled for output using a Multiplier of 0.01 and Offset of 0.0; a value of "90" in the output map is equal to 0.90 m depth calculated by the model.

Any software capable of opening or importing generic/raw binary spatial arrays can be used to analyze and/or animate the time series of output maps. The Open Source GRASS GIS and its associated "xganim" animation program can be used to analyze and visualize the output. As reviewed in the Appendix of this Chapter, many other tools, such as the Open Source OpenDX, or the commercial IDL, are available for geospatial analysis and visualization. The ELM developers have various postprocessing codes (using a custom C program, GRASS, and IDL scripts) for summarizing and visualizing spatial output, but they are not fully developed for public release.

10.6.6 Spatial: Point (grid cell) time series

The "PtSer/" directory contains tab-delimited text files with point (grid cell) time series output. Virtually any variable in the model may be output in this format, at user-selected grid cell locations and output intervals (see Runtime configuration section of this Chapter). A separate file is created for each model variable that is requested for output, and each file has multiple fields (columns) for multiple grid cell locations.

10.6.6.1 Postprocessing point time series text files

All point time series files are in tab-delimited text format, and thus can be directly read into any spreadsheet program such as Open Office Calc or Microsoft Excel. The primary method for ELM postprocessing is the use of scripts written in the Python scripting language. The ELM developers have a flexible set of Python postprocessing scripts that will produce a variety of summaries of these data for visualization and analysis, but that development is not complete enough for release. Spreadsheet templates for different summaries of the output data are available from the developers, but are unsupported.

10.6.7 Spatial: Canal (vector) time series

The "Canal/" directory contains tab-delimited text files with canal (vector) time series output of

- CanalOut: instantaneous water depth in all canal reaches,
- CanalOut_P: instantaneous total phosphorus concentration in all canal reaches,
- CanalOut_S: instantaneous salt/tracer concentration in all canal reaches.

These variables are all of the state variables used in the canals of water management simulation, and the user can select the output interval for this group of outputs (see Runtime configuration section of this Chapter).

10.6.7.1 Postprocessing canal time series text files

All canal (and water control structure) time series files are in tab-delimited text format, and thus can be directly read into any spreadsheet program such as Open Office Calc or Microsoft Excel. The primary method for ELM postprocessing is the use of scripts written in the Python scripting language. The ELM developers have a flexible set of Python postprocessing scripts that will produce a variety of summaries of these data for visualization and analysis, but that development is not complete enough for release. Spreadsheet templates for different summaries of the output data are available from the developers, but are unsupported.

10.6.8 Spatial: Structure (point/cell) flow time series

The "Canal/" directory contains tab-delimited text files with water control structure (vector) time series output of

- structsOut: summed (across each output interval) water flows through all water control structures,
- structsOut_P: flow-weighted (across each output interval) mean total phosphorus concentration at all water control structures, and
- structsOut_S: flow-weighted (across each output interval) mean salt/tracer concentration at all water control structures.

These variables are all of the state variables used in the structure flows of water management simulation, and the user can select the output interval for this group of outputs (see Runtime configuration section of this Chapter).

10.6.8.1 Postprocessing structure time series text files

All water control structure (and canal) time series files are in tab-delimited text format, and thus can be directly read into any spreadsheet program such as Open Office Calc or Microsoft Excel. The primary method for ELM postprocessing is the use of scripts written in the Python scripting language. The ELM developers have a flexible set of Python postprocessing scripts that will produce a variety of summaries of these data for visualization and analysis, but that development is not complete enough for release. Spreadsheet templates for different summaries of the output data are available from the developers, but are unsupported.

10.7 Advanced applications

The following topics are generally beyond the scope of this User's Guide Chapter, but are included in brief summary in order that users may have some guidance if they desire to advance beyond standard, historical simulation runs.

10.7.1 Sensitivity analysis

The user can run the automated sensitivity analysis on model parameters whose results were was described in the Uncertainty Chapter. The "S_ParmName" parameter in the Driver.parm configuration file (see Model configuration section of this Chapter) is used to control which parameters are modified as follows:

- S_ParmName= ALL: evaluate model sensitivity to changes in each of the parameters listed in the input data file SensiParm_list,
- S_ParmName= ParameterName: evaluate model sensitivity to changes in the single parameter whose name is ParameterName, or
- S_ParmName= NONE: no sensitivity analysis, and thus a normal, single simulation run using only the nominal parameter sets

The values of the parameter ranges are changed in the GlobalParms and the HabParms databases: separate "worksheets" are available to calculate and export _LO and _HI (low and high estimates of parameters in) parameter files that are read by the model during the sensitivity analysis. Upon invoking a sensitivity analysis via the S_ParmName parameter, a suite of simulations are executed sequentially when the user executes the model (from either the Run or the go script). An Open Office Calc template is available from the ELM developers for postprocessing the single output file[9] from the multiple runs.

10.7.2 Evaluating project alternatives

To evaluate most (likely all) water management alternative scenarios, no source code needs to be changed, and ecological parameters (in GlobalParms and HabParms databases) generally are not expected to be changed. For a new management alternative, the user just needs to modify the following input data files (which are all described in the Data Chapter):

- CanalData.chan: any changes to the canal/levee topology and attributes,
- CanalData.struct: any changes to the water control structure attributes,
- CanalData.struct_wat: water control structure (daily) water flows (that are output from SFWMM or other tool),

[9] actually, the single BIRavg output file for all of the sequential simulations can be spread over multiple files (unrelated to sensitivity) if the number of Indicator Regions is large, i.e., BIRavg1 – BIRavg5 as described in the Model output section of this User's Guide Chapter

- CanalData.struct_TP: water control structure (daily) Total Phosphorus concentrations,
- CanalData.struct_TS: water control structure (daily) Total Salt/tracer concentrations.
- (?) GlobalParms_NOM: if appropriate, alter the parameter that estimates the annual rate of sea level rise

To add a new canal, a new canal reach ID is added to the CanalData.chan text file, adding the canal reach attributes and the geographic point coordinates that define the segments of a reach. Existing canal reaches can be “turned off” (ignored by model) by assigning a negative width attribute to that reach. GRASS scripts are used to aid in this process and visualize any new topology of the canal network. Other scripts are used to determine which, if any, new water control structures are required, extracting the appropriate time series of flows from a “DSS” formatted file that was output from the SFWMM (which is the current modeling tool for evaluating hydrology of management alternatives).

The meteorological boundary conditions for the 1965-2000 period of record are contained in the current (rain.BIN, ETp.BIN) input files. The general assumption in forecasting the responses of the system to management changes is the following: If the system were to be subjected to the same meteorological conditions as those observed between 1965-2000, how would the system respond under a new suite of management rules and/or infrastructure?

Obviously (?), there are other assumptions that are involved with forecasting the system responses to future management alternatives. While the data modification/input methods are generally simple and scripted, the details of the steps, including the assumptions and the necessary data quality assurance, are beyond the scope of this User’s Guide.

10.7.3 New subregional applications

To implement a new subregional application of the ELM, no source code needs to be changed. The following input files require modification/re-scaling:

- Input maps: all input maps must be reconciled to the spatial resolution and extent of the new domain (i.e., with new data, or rescaling/interpolating existing data)
- CanalData.chan: canal reaches from the regional model application that are within the new domain may be kept (as they use geographic, not grid cell, coordinates); the upper left corner of the origin of the rectangular domain requires changing (if necessary),
- CanalData.struct*: water control structure attributes and flow/concentration data from the regional model application that are within the new domain may be kept, but the Structs_attr.fmp database (or another calculator) should be used to calculate the new grid cell locations of the geographic coordinates of the water control structures; unused structures need to be removed from all CanalData.struct* files,
- Driver.parm: modify the parameter that defines the model grid cell area

- Model.outList: use the ModelOutlist_creator interface to calculate the new model grid cell locations of the named monitoring stations for which output is desired
- gridmapping.txt: run the GridMap preprocessor application to generate the new linked list of the SFWMM grid cells that are mapped to the grid cells of the new ELM application (for boundary condition data on meteorological inputs and stage at the periphery of the new domain)

While the data modification/input methods are generally simple, the details of the steps, including the necessary data quality assurance, are beyond the scope of this User’s Guide.

10.8 Appendix

10.8.1 Driver.parm configuration file

The following table contains extended documentation of all of the adjustable parameters in the "Driver.parm" input file that is input to the model to configure a simulation run.

Parameter	Brief metadata	Extended instructions
/MyOutputPath/	{output path (absolute path, w/o ProjName) }	Path for model output can be on any file system. If user requests many animations at high output frequency (e.g., 20 variables, daily), a local hard disk directly attached to host machine can become important to model run time.
1/1/1981	{Sim start date (yyyy/mm/dd), min= 1965/01/01 }	User is informed of error if attempting to start simulation outside of the range of available boundary condition data (1/1/1981 or 1/1/1965 through 12/31/2000, depending on project).
12/31/2000	{Sim end date (yyyy/mm/dd), max= 2000/12/31 }	User is informed of error if attempting to end simulation outside of the range of available boundary condition data (1/1/1981 or 1/1/1965 through 12/31/2000, depending on project).
00/00	{Sim re-init date (mm/dd)(no Position Analysis, mo=00)}	Used only in "Position Analysis", in which simulation is re-initialized annually on a given month/day. If month=00, Position Analysis is not invoked. Position Analysis is not fully updated/supported in v2.5.
ELM	{model name (needs to match CanalData input files)}	Used in distinguishing subregional model projects (e.g., ELM_wca2@500m) from the default regional "ELM". Used primarily to ensure model is using correctly geo-referenced data in CanalData.* input files in subregional projects.
Model version= v.2.5	{model version (e.g., v.2.1)}	Model version identifier to label output files.
CellArea= 1000000.0	{grid cell area, m^2}	The area of an individual model grid cell; standard regional application is 1,000,000 m^2 (1 km^2).
budg_Intvl= 0.0	{interval (julian days), BIR stats (0=calendar-month)}	Time interval for summary calculations in all budget output files (./Budget/budg_*) in Basins/Indicator Regions (BIR). Value >0 is julian day interval; a value=0.0 is an exact calendar-month interval (accounting for leap years etc.).
avg_Intvl= 30.0	{interval (julian days), cell-avgs (0=calendar-month)}	Time interval for all internally-calculated temporal means in BIRavg output files (./Budget/BIRavg*) in Basins/Indicator Regions (BIR). Value >0 is julian day interval; a value=0.0 is an exact calendar-month interval (accounting for leap years etc.).
seed= 568	{random number seed; UNUSED in current	UNUSED

	version}	
dt= 1.0	{time step (days, use 1.0) for vertical fluxes}	The model time step for vertical solutions only. The dt should remain at 1 day for any scale application.
hyd_iter= 12	{**EVEN number**, number of horiz iterations per dt}	The number of iterations, or time slices, per dt for horizontal solutions such as cell-cell overland flow. To determine the appropriate value for a new application, see the ELM documentation for theoretical estimates for different model scales and expected velocities. The 1 km^2 regional ELM application uses hyd_iter = 12 (i.e., a 2 hour time step).
debug= 2	{0:Minimal 1:BasinChek 2:Default 3:More 4:Canal 5:Lots}	The choice of how much information to print to a debug (text) output file (./Debug/DriverX.out, X'th simulation, X=1 in a standard run w/o Sensitivity Analysis). The recommended standard is debug= 2. Higher values will produce very large volumes of information and should be used in relatively short simulations. **See text below this Table for details.
debug_point= 62 43	{focal cell (row col) for Driver1.out if debug>2}	The row-column coordinates of the focal grid cell for 5x5-cell windows of output data that are written to the (text) debug file at high values of the debug parameter.
S_ParmName=NONE	{Sensitivity analysis: "NONE", "ALL", or ParameterName}	Invoke an automated sensitivity analysis on "ALL" parameters in the input data file "SensiParm_list", or on a single parameter whose exact name is provided, or "NONE" for a standard, single simulation run. See text of User's Guide for details.
HabSwitchOn= 0	{Habitat switching (succession) on=1, off=0}	Invoke the habitat switching (succession) module of the model. See text of Model Structure Chapter in the ELM documentation for some details on module.
WatMgmtOn= 1	{Water management and canal network on=1, off=0}	Invoke the water management modules, with flows through water control structures in the network of canal/levee vectors. Normally this is "on". If turned "off", all water management network topologies and managed flow dynamics are inoperative, and thus the only flow constraints are those imposed along the periphery of the model domain (aka a simulation of the "Natural System" that is not compartmentalized).
Scenario= calib	{scenario/alternative name (case sensitive)}	Model scenario (alternative) identifier to label output files.
Scenario modifier= myRun	{scenario/alt modifier or descriptor}	An additional descriptor of specifics to add to the model scenario (alternative) identifier to label output files.

Sectors= 1 0 7 10 9 2 8 12 4 99;		The (left-to-right) sequence of calls to ecological modules (sectors) in the time loop of the simulation. See text of Model Structure Chapter in the ELM documentation for details on the structure of the model time loop, and summaries & details of each module. A single-phrase description of each module is given below in this table (and the "Driver.parm" file).
{Below are not input fields; for descriptive purposes only}		
Sequence for calling modules:	1 0 7 10 [13] 9 2 8 12 4 99	Recommended sequence of module calls. See text of Model Structure Chapter in the ELM documentation for details on the structure of the model time loop.
Module #0	hydrology: horiz raster fluxes (& water management if it is on)	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #1	global forcings: vertical fluxes (& succession if it is on)	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #2	algae/periphyton: vertical fluxes	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #4	DOM/DOP: vertical fluxes	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #7	hydrology: vertical fluxes	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #8	macrophytes: vertical fluxes	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #9	phosphorus: vertical fluxes	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #10	salt/tracer: vertical fluxes	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #12	Floc: vertical fluxes	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #13	ESP P settling model mode, do NOT invoke 2,4,8,9,12	See text of Model Structure Chapter in the ELM documentation for details on module.
Module #99	summary budget & stats	See text of Model Structure Chapter in the ELM documentation for details on module.

*10.8.1.1 **Debug levels:*

- debug =0 Echo short console info on iteration# etc, print critical error/warning info. USE WITH CAUTION.
- debug =1 Report mis-configured basin flows. Currently same level as debug=2.
- debug =2 DEFAULT for general use, more warnings etc.
- debug =3 Echo long console output, prints additional (non-critical) errors/warnings to DriverX.out (for X'th simulation run) file
- debug =4 Prints details of cell vertical and/or horizontal flux data, and details of indiv canal fluxes, to DriverX.out (for X'th simulation run)
- debug =5 Prints grid_map information, and prints to another canal debugging file for special purposes

10.8.2 Environment variables

The required environment variables are the following:

Environment variable	Unix path	Description
ELM_HOME	/My/Directory/	The absolute path to the "home" directory where you install the source code (and by default, the data of multiple projects) of ELM. Can be anywhere on the user's networked file system(s).
ELM_ARCHIVE_PATH	/Any/Directory/arc_out/	The absolute path to the directory where simulation run "keepers" of (multiple) ELM project(s) are archived (and thus not overwritten in subsequent simulation runs!). Can be anywhere on the user's networked file system(s). Suggested default during ELM installation was within the $ELM_HOME.

The highly recommended addition to the user's path (to executables) is:

Add to user's path env.	Description
$ELM_HOME/SME/scripts/	The location of all ELM scripts.

The optional environment variable is the following:

Environment variable	Unix path	Description
ModelPath	/Anywhere /SME/Projects/	The absolute path to the (multiple) project(s) of ELM data and executables. Can be anywhere on the user's networked file system(s). For testing different code sets with one single data location, we can set the $ModelPath as a system environment variable. In the default (distribution) version, the $ModelPath is determined from $ELM_HOME and is not needed as an environment variable.

10.8.3 Directory/file structure

The complete directory structure of an ELM project.

Directory structure	File type	File descriptions
$ELM_HOME/		
include/sme/	source code	header files
SME/		
scripts/	source code	unix shell scripts
SMDriver/Sources/		
Driver_Sources/	source code	main program, utilities
SpatMod/	source code	spatial fluxes
Tools/	source code	I/O tools
UnitMod/	source code	unit model
Dbases/	databases	databases for data export to model
Projects/		
ELM2.5/		
Data/	input data	all input data files (maps in subdirs)
Map_bin/	input data	all map binary arrays
Map_cats/	input data	all map category definitions
Map_head/	input data	all map header definitions
Map_hist/	input data	all map metadata/history
RunParms/	input data	runtime configuration parameters
Load/	executable	compiled model executable
Output/ [1]		
Animation1...60/	output data	multiple directories to hold map outputs
Budget/	output data	budgets and preset Performance Measures
Canal/	output data	canals and structures
Debug/	output data	debug-related
PtSer/	output data	point (cell) time series
$ELM_ARCHIVE_PATH/		
ELM2.5/		
MyFirstRun/		
VarNameA	output data	archived map output of VarNameA
VarNameB	output data	archived map output of VarNameB
VarNameXYZ	output data	archived map output of VarNameXYZ
Budget/	output data	archived budget and preset PMs
Canal/	output data	archived canal and structure summaries
Debug/	output data	archived debug-related files
PtSer/	output data	archived point (cell) time series
Input/	input data	archived input data (subset, parameter files)

[1] Output directory may be anywhere, including outside of $ELM_HOME

10.8.4 Software recommendations

In order to interpret input and output data, it is recommended that the user at least has access to the Open Source software of the GRASS GIS and the Open Office Calc spreadsheet system. Both are available as pre-compiled binaries for a number of computing platforms, and thus are very simply installed.

10.8.4.1 Geographic Information System (GIS)

The GRASS[10] GIS can be used to analyze model input and output data. GRASS excels in raster data processing and analysis, with many useful functions for landscape analysis. It also fully supports the vector (canal) and point (water control structures, monitoring locations) data required for ELM. Through the use of unix symbolic links between the GRASS and the ELM data directories, the ELM directly reads GRASS project data files (uncompressed binary data and text header) as model input. However, no GRASS-specific encoding of binary information is used, and thus the data files may be opened with any program that can read binary data arrays. Scripts are available to directly input and visualize the ELM canal vectors in GRASS. Other GIS and/or spatial mapping software tools can serve similar purposes.

10.8.4.2 Animated visualization

The GRASS GIS and its associated "xganim" animation program can be used to visualize animations of the output. We also use other tools, such as the Open Source OpenDX[11] and IDL[12] for such purposes, as both have advanced functionality relative to xganim.

10.8.4.3 Data management

While MySQL[13] is our targeted relational database system, we currently use the functionality of spreadsheet data systems in Open Office Calc[14] (which is fully compatible with Microsoft Excel). FileMaker Pro[15] has been used for a relational database system for parts of ELM, but will be entirely phased out with MySQL in the future.

10.8.4.4 Advanced scripting

Python[16] (and an associated graphics library PyChart[17]) is our choice for developing object-oriented, advanced script applications for post-processing the model and other tasks.

[10] http://grass.itc.it/ (Open Source)
[11] http://www.opendx.org/ (Open Source)
[12] http://www.rsinc.com/ (commercial)
[13] http://www.mysql.com/ (Open Source)
[14] http://www.openoffice.org/product/calc.html (Open Source)
[15] http://www.filemaker.com/ (commercial) 30-day trial version of the software
[16] http://www.python.org/ (Open Source)
[17] http://home.gna.org/pychart/ (Open Source)