Appendix B

Prepared by the ESHMC On-Farm Subcommittee

and

Ben Britton (IDWR IT Programmer)

Contributors in alphabetical order

Jim Brannon, Ben Britton, Willem Schreüder, Greg Sullivan, and Allan Wylie

Contents

Introduction

This Appendix is a compilation of reviews of the **mkmod** code. **Mkmod** was written by Willem Schreüder (Principia Mathematica) with assistance from Greg Sullivan (Spronk Water Engineers). **mkmod** calculates the ESPAM2.1 water budget and produces a MODFLOW "WEL" file. During this process it implements what is known as the "On-Farm algorithm" partitioning irrigation water between canal seepage, satisfying the crop irrigation requirement, deep percolation, and field runoff. The On-Farm algorithm was developed by Greg Sullivan and adopted by the Eastern Snake Hydrologic Modeling Committee (ESHMC) for use in the Enhanced Snake Plain Aquifer Model (ESPAM).

This document includes a description of the On-Farm algorithm by Greg Sullivan and an overview of **mkmod** by Willem Schreüder. The overview is derived from a presentation to the ESHMC during the August 2011 MKMOD/MODFLOW training. This presentation is mentioned in the Britton review and can be found on the web at 'http://www.idwr.idaho.gov/Browse/WaterInfo/ESPAM/meetings/MKMOD_MODFLOWTraining/'

This document also includes three reviews of the **mkmod** code. The first review is by Jim Brannon (Leonard Rice Engineers). Brannon agreed to provide a peer review and present his results to the ESHMC. The Brannon review is focused on an analysis of the On-Farm algorithm. The second review is by Ben Britton (Idaho Department of Water Resources, IDWR). Britton translated the program from PERL to English. The third review is a comparison with hand calculations performed by Allan Wylie (IDWR).

mkmod represents a paradigm shift in ESPA modeling. On mixed source lands, the ESPAM1.1 algorithm assumes that EvapoTranspiration (ET) estimates are reliable and extracted ground water if surface water diversions do not meet the crop irrigation requirement. On the same mixed source lands, the On-Farm algorithm assumes that the diversion data are reliable, and imposes deficit irrigation if diversions do not meet the crop irrigation requirement. The results of this change in philosophy are:

1) If ET estimates are too high or surface water diversions are too low, the ESPAM 1.1 algorithm overestimates ground water withdrawals and underestimates recharge. **mkmod** can compensate through adjustments in some of its parameters.
2) With an underestimate of surface water irrigated land and an overestimate of ground water irrigated land, ESPAM1.1 will allow too much deep percolation in the surface water irrigated land and too much pumping on the ground water irrigated land, resulting in a local distortion in the spatial distribution of recharge and pumping. The same underestimate of surface water irrigated land with **mkmod** results in too much deep percolation on the surface water irrigated land and deficit irrigation on the ground water irrigated land. This results in changes in the consumptive use, and a distortion in the water budget for the stress period(s) in question.

3) With an overestimate of canal seepage, the ESPAM1.1 algorithm will pump additional ground water to meet the crop demand, offsetting the overestimate of canal seepage with an overestimate of pumping, resulting in a local distortion in the water budget. The same overestimate in **mkmod** results in deficit irrigation and a corresponding change in consumptive use resulting in a distortion in the water budget.
4) In the absence of mixed source land, if surface water shortages ocur the ESPAM1.1 algorithm requires manual adjustment to avoid underestimation of recharge. **Mkmod** parameters can be set so the appropriate reduction in consumptive use occurs automatically.
5) Both algorithms redistribute recharge to the service area if canal seepage is underestimated.
6) Both algorithms will overestimate recharge if ET estimates are underestimated.
7) Imprecision in return-flow estimates similarly affect both methods.

The shortcomings associated with **mkmod** resulting in deficit irrigation where it did not occur, can be compensated for by using the output files to identify the specific entities and stress periods where deficit irrigation is occurring in the model. Once spatially and temporally identified, the sources of most mistakes can usually be repaired.

During the March/April 2009 meeting, when **mkmod** was first introduced to the ESHMC, Allan Wylie was asked by the ESHMC to compare the output of **mkmod** in ESPAM1.1 mode with output from **readinp** (the ESPAM1.1 water budge program). Wylie reported to the committee in an e-mail dated June 26, 2009 (this e-mail is posted in the July 2009 meeting directory) that **mkmod** yielded comparable results to **readimp**.

Mkmod was updated many times during calibration. Most changes were minor, some required changes to input files, some required changes to information included in one or more of the output files, and some improved program efficiency and were transparent to the user. Brannon mentions transitions between **mkmod**4, **mkmod**5, and **mkmod8**. The Britton review mentions **mkmod8.1.** Most of the changes between **mkmod**4 and **mkmod**5 altered the method **mkmod** used to read several of its input files. It also resulted in some minor changes in some output files. **Mkmod**8 fixed a bug in soil moisture accounting. **Mkmod8.1**, the version used in calibration of ESPAM2.1, optionally outputs a *.rfl file that can be used for debugging, or to assist in model calibration.

The following description of the On-Farm Algorithm was prepared by Greg Sullivan and is included here with minor editorial changes.

Description of the On-Farm Algorithm

A portion of the irrigation water delivered to a farm is consumed by crops through evapotranspiration. The unconsumed water will percolate below the root zone of the crop and recharge the aquifer, and in some cases will flow off the irrigated field as surface runoff. The losses to deep percolation and surface runoff will vary based on the amount of irrigation water applied, the method of irrigation, irrigation management practices, and other factors.

All crops extract and transpire water from the soil column within a root zone that typically extends two to five feet below the surface depending on the crop. In arid areas such as Idaho, water enters the soil column primarily through irrigation, and to a lesser extent from precipitation. The purpose of irrigation is to refill the soil column within the root zone of the crop after it has been depleted over a number of days or weeks by the crop evapotranspiration processes.

Except under severe deficit irrigation, it is not physically possible to refill the root zone without loss of water to deep percolation below the root zone, and in some cases, surface runoff. This is due to the non-uniformity of soils, variations in root depths, and imperfect irrigation application practices.

The maximum irrigation efficiency represents the reasonable upper limit of the amount of applied irrigation water, expressed as a percentage of farm delivery, that can be delivered to the root zone of the crop either for immediate use, or stored for later use. The maximum irrigation efficiency will vary depending on the method of application. The two primary methods of irrigation water application in Idaho are gravity irrigation (flood and furrow application) and sprinkler irrigation.

Sprinkler irrigation is typically more efficient than gravity irrigation because water can be applied more uniformly with sprinklers. Properly managed sprinkler irrigation will have minimal surface runoff, but invariably will result in some deep percolation due to the non-uniform soils and root depths across an irrigated field. In order to fully irrigate all portions of a field, there will be portions of the field that are over-irrigated (e.g., in areas with shallower roots or lower water holding capacities).

Under gravity irrigation, water is delivered to the upper end of a sloped field and left running until the water has infiltrated the soil and filled the root zone. Due to the time that it takes for the irrigation water to reach the lower end of the field, it is necessary to over-irrigate the upper end of the field in order to fully refill the root zone at the lower end of the field. In addition, there typically will be surface runoff from the lower end of a gravity irrigated field.

An On-Farm water budget algorithm was developed to compute the irrigation consumptive use, deep percolation, and surface runoff from each model cell with surface water irrigation. The algorithm computes the crop water consumption as the lesser of (a) the crop irrigation water requirement, and (b) the available irrigation supply limited by a specified maximum irrigation efficiency depending on the method of irrigation application.

The irrigation water losses to surface runoff and/or deep percolation are computed as the sum of (a) the inefficient portion of the irrigation application, and (b) the irrigation application that is in excess of the crop water requirement. The irrigation losses are divided between deep percolation and surface runoff based on user specified parameters.

The following is the On-Farm water budget algorithm for computing deep percolation recharge to the aquifer and surface runoff.

Recharge

Rech = Initial recharge from inefficient portion of irrigation + recharge from excess application

Rech = (1 – OFE) x Dh x DPin + Max (Peff + OFE x Dh – ET x A – Max(ΔSm,0), 0) x DPex

Surface Runoff

SRO = Initial runoff from inefficient portion of irrigation + runoff from excess application

SRO = (1 – OFE) x Dh x (1 – Dpin) + Max (Peff + OFE x Dh – ET x A – Max(ΔSm,0), 0) x (1 – DPex)

Where

Peff = effective precipitation

OFE = maximum On-Farm efficiency

Dh = farm headgate delivery

A = ET adjustment factor

DPin = portion of initial loss to deep percolation

DPex = portion of excess delivery to deep percolation

ΔSm = increase in soil moisture

Based on discussion among the ESHMC members, the maximum On-Farm irrigation efficiency was set at 0.85 for sprinkler irrigation and 0.80 for gravity irrigation. The deep percolation factors (DPin and DPex) were determined during model calibration.

Schreüder Overview

The following description of **mkmod** was prepared by Willem Schreüder and is included here with minor editorial changes.

Introduction

The **mkmod** program is used to build MODFLOW input files for the ESPAM model verion 2.1. The name is a contraction of MaKe MODflow files.

It replaces the **readinp.for** program used with the ESPAM1.x models. The **readinp.for** program served as the tail end of the recharge tool to assemble different inputs into files that can be read by MODFLOW. Similarly, **mkmod** reads various input files and produces output files in MODFLOW format which are then used to do the actual model runs.

The **mkmod** program can be run in Version 1.1 mode where it replicates the **readinp.for** functionality. However, **mkmod** also adds three new features not present in the **readinp.for,** namely the On-Farm algorithm, soil moisture accounting, and the ability to calculate surface returns based on the On-Farm algorithm, instead of specifying the returns. In addition, **mkmod** can also calculate an initial steady-state stress period as the average of a user selected set of stress periods and produce separate stress files for recharge, pumping and similar stresses which can be used to calculate detailed budgets from the MODFLOW output. The **mkmod** program also produces detailed summary tables of the calculated stresses by entity and other groupings.

The **mkmod** program operates at the entity and model-cell level. It primarily maps inputs specified by model entity to individual cells for use in MODFLOW. Most calculations are done at the model-cell level. The time increments by **mkmod** correspond to stress periods in MODFLOW. In ESPAM2.1 the time increments are calendar months.

The prime reason for creating the **mkmod** program was to implement the On-Farm algorithm. The On-Farm algorithm was proposed by Greg Sullivan, a member of the Eastern Snake Hydrologic Modeling Committee (ESHMC). The implementation of **mkmod** was performed by Willem Schreüder (ESHMC member). The **mkmod** code was peer reviewed by Jim Brannon (ESHMC member) and Ben Britton (an IDWR IT systems programmer). User testing was done by Allan Wylie and Jennifer Sukow (both ESHMC members).

The **mkmod** program is run from the command line. Command line options are used to control which algorithm is used, e.g. the Version 1.1 or On-Farm algorithm, whether soil moisture is simulated, and so on. The **mkmod** control file (*.mdl file) controls settings such as the model dimensions, units, stress periods, as well as definitions such as the types of off-site pumping. In addition, **mkmod** reads data files from other tools that pre-process the data, as well as files specifying data by entity such as the irrigation efficiencies required by the On-Farm algorithm.

Outputs produced by the **mkmod** program are primarily MODFLOW input files. This may consist of a net recharge file, also known as the *well term* in ESPAM parlance, which is in the MODFLOW well file format. Alternatively the stesses may be saved as individual budget terms in MODFLOW recharge or well file formats. The **mkmod** program also produces a detailed summary of budget terms by stress period and entity, as well as summaries for surface water, groundwater and all other entities. This output file is saved in HTML format, and can be viewed in a web browser or spreadsheet program. The **mkmod** can also produce additional diagnostic outputs, such as the acreage by cell over the simulation. The *.mdl file also allows user specified summaries to be generated of any budget term and saved as a text file. This is a convenient way of summarizing input budget terms as part of the calibration process.

mkmod operation

The **mkmod** program operation consists of two main phases. The first phase operates on data that do not change over the course of the simulation. This includes definitions such as active model cells

and soil types. The second phase operates on data that are supplied for every stress period, which corresponds to calendar months in ESPAM2.1.

For every stress period, the **mkmod** program performs the following operations:

1. Read data for current period
2. Calculate irrigated recharge/pumping
3. Calculate non-irrigated recharge
4. Distribute canal seepage
5. Distribute tributary underflow and perched river seepage
6. Save data

Calculating recharge and pumping

The most complex step in this calculation is step 2, where irrigated recharge and pumping is calculated. The algorithm for this step is as follows:

- Calculate the applied water by entity as *Diversion – Canal Seepage + Off-site pumping*
- Loop over entities
  - Loop over gravity and sprinkler lands
    - Loop over cells
      - Calculate Crop Irrigation Requirement *(CIR)*, *pumping*, *recharge* and *change in soil moisture*
      - Accumulate irrigated acres per cell

The key step in this algorithm is the step that calculates the CIR, pumping, recharge and change in soil moisture for every cell. The specific algorithm applied depends on the command-line switches selected. In addition, groundwater and surface water are treated differently. To simplify the expressions, the calculations are done on a unit area bases, and then multiplied by the acres to get a volume. All calculations in **mkmod** are based on totals for the period, typically volumes. This makes accumulation of multiple stress periods a simple addition.

The CIR is calculated for every cell under each entity. The *Adjusted ET* is defined as

*Adjusted ET = ETadj * CellET*

where the *ETadj* is the ET adjustment specified for this entity by gravity or sprinkler, and *CellET* is the ET read for this cell and period from the ET input file. CIR is then calculated as

CIR = Adjusted ET – Precip

for every cell where *Precip* is precipitation and is read for this cell and stress period from the precip input file.

For groundwater entities, when the CIR is negative, pumping is set to zero. If the CIR is positive, pumping is set to

Pumping = CIR / Efficiency

where the efficiency is the user specified irrigation efficiency by sprinkler or gravity for this entity.

Finally rechage is calculated as

Recharge = *Pumping* – *CIR*.

Note when CIR is negative, pumping will be zero, and recharge will equal -CIR. Net recharge will always be

Net Recharge = -*CIR* = *Recharge* – *Pumping*

and

Recharge = *Precip* + (1-*Efficiency*) x *Pumping*.

Also note that soil moisture is never used in this calculation as it is asserted that for these groundwater entities, pumping will be used to fully supply the CIR, and no deficit irrigation will occur.

For surface water entities, the On-Farm algorithm is applied by default. The first step is to determine the amount of over or under irrigation. The net application is defined as

Net Applied = *Efficiency* x *Applied* – *CIR*

where *Applied* is diverted volume less canal loss plus off-site pumping, and *Efficiency* is the irrigation efficiency by sprinkler or gravity for this entity. When the *Net Applied* is positive, the *Excess* is set equal to the *Net Applied* and the *Deficit* is zero. When the *Net Applied* is negative, the *Excess* is set equal to zero, and the deficit is – *Net Applied*.

By default, **mkmod** will adjust any excess or deficit for soil moisture. The capacity of the soil profile to store water is called the soil moisture sink, while the capacity of soil moisture to supply water to the plant is called the soil moisture source. These quantities are defined as

Soil Moisture Sink = *Depth* x (*Field Capacity* – *Previous Soil Moisture Content*),

Soil Moisture Source = *Depth* x (*Previous Soil Moisture Content* – *Wilting Point*).

The rooting depth, field capacity, and wilting point are specified for each entity.

When there is a deficit and the soil moisture source is positive, the deficit will be reduced by the minimum of the deficit and the soil moisture source, and the soil moisture will be reduced by a corresponding amount. Note that the soil moisture can never be reduced below the wilting point using this procedure. Also note that if the soil moisture is insufficient, the deficit may remain positive.

If there is an excess, and the soil moisture sink is positive, the excess will be reduced by the minimum of the excess and the soil moisture sink, and the soil moisture will be increased by a corresponding amount. Note that the soil moisture can never exceed the field capacity using this procedure. Also note that some excess may remain if the soil moisture sink is less than the initial excess.

After the soil moisture adjustment, recharge and runoff are calculated as

Recharge = *DPin* x (1-*Efficiency*) x *Applied* + *DPex* x *Excess*

Runoff = (1-*DPin*) x (1-*Efficiency*) x *Applied* + (1-*DPex*) x *Excess*.

where *DPin* is the initial deep percolcation fraction, and *DPex* is the excess deep percolation fraction. In essence, the *DPin* fraction partitions the inefficient portion of the overall irrigation

application between recharge and runoff, while *DPex* partitions any excess irrigation between recharge and runoff.

Note that when there is deficit irrigation (after soil moisture), the crop is shorted and ET is less than that specified. Under such deficit conditions, the specified efficiency is the maximum efficiency that can be achieved. When there is any excess, the actual efficiency would be less than this efficiency because any additional applied water would result in recharge or runoff.

Alternatively, recharge for surface water entities may be calculated using the ESPAM 1.x algorithm. This algorithm calculates recharge as

Recharge = Applied – CIR.

Note that this algorithm does not calculate any runoff. Instead, the user is required to specify the amount of runoff/returns, which is subtracted from the diversion to give the amount of applied water. The maximum application efficiency in this algorithm can be 100%, and the recharge may be negative in the case of deficit irrigation, which requires the user to manually zero out these values.

When the *Excess* is positive, the On-Farm and ESPAM 1.x algorithms would yield the same result, except that the *DPin* and *DPex* parameters partition the overage between recharge and runoff.

In summary, for groundwater entities, **mkmod** uses pumping to meet the CIR. The soil moisture remains constant, and the net pumping equals the adjusted ET minus precipitation. For surface water entities, the **mkmod** default is to use the On-Farm algorithm and soil moisture. The applied water is the surface diversion minus ditch leakage plus off-site pumping. Any excess or deficit irrigation will be mimized by soil moisture up to the field capacity or wilting point, respectively. Recharge is a fixed fraction of the amount of applied water and a fixed fraction of any excess. Similarly runoff is a fixed fraction of the amount of applied water and a fixed fraction of any excess. When deficits occur, ET is shorted.

All these recharge, pumping and runoff calculations are done cell by cell, for each entity, and for sprinkler or gravity irrigation.

Calculating other recharge components

By comparison to the calculation of pumping, recharge and runoff for irrigated lands, the remainder of the stresses are relatively simple. These calculations mostly just distribute volumes calculated by other programs to individual model cells.

Non-irrigated recharge is calculated for all active model cells where irrigated lands do not cover the entire cell. The non-irrigated acres for each cell are calculated by subtracting the acres irrigated by various surface and groundwater entities from the total cell area. The non-irrigated acres cannot be less than zero.

For each cell, the non-irrigated recharge is calculated as

Recharge = *Soil Factor* x *NIR* x *Area*

where *NIR* is the non-irrigated recharge depth by cell for this period, *Area* is the area not irrigated, and the *Soil Factor* is a multiplier based on the soil type which is specified by cell.

Canal seepage is specified as a volume by entity and stress period. In addition a scale factor is used to adjust the amount for each entity. The location where the seepage should occur is specified as a list of model cells. The entity total is distributed proportionally to the specified cells.

Tributary underflow and perched river seepage are similarly specified as volumes by reach and stress period. In addition, a scale factor is used to adjust this amount for each reach. A list of cells comprising the reach is specified by the user, and the scaled volume is distributed proportionally to the cells corresponding to that reach.

The fixed point and off-site pumping terms allow additional flux to be added to the budget. Fixed point terms include wetlands, urban, exchange and Mud Lake pumping, and the user specifies the budget item and model cell to be adjusted. For every stress period, the corresponding budget item is then adjusted by the specified volume in the corresponding cell. Off-site pumping always adjusts the pumping term, and is added to the applied water for the corresponding entity.

Running mkmod

The **mkmod** program is written in the Perl computer programming language. This language was chosen because of its expressive power. Features such as arrays indexed by strings allow the **mkmod** program to express concepts such as *loop over the following entities* by using the actual names of the entites instead of a numerical index. This simplifies the code and reduces the chance of programming errors.

The Perl language is interpreted not compiled. It therefore needs a Perl interpreter to be installed on the computer where **mkmod** is being run. On Unix, Linux and OS/X systems, Perl is part of the standard installation, and can be run as

mkmod <options and parameters>

On Windows based computers, Perl is typically not installed, and due to the way Windows programs are launched, must be run as

perl mkmod <options and parameters>

Alternatively the Perl interpreter can be compiled with the **mkmod** program into an executable program using the PAR-packer utilty, in which case the resulting executable **mkmod.exe** can be run just like on other systems.

A typical **mkmod** run would be

mkmod -ss E110712A

This runs **mkmod** using files named E110712A.*. There are a number of input files with a file root of E110712A that would be read. The **-ss** flag omits the steady state stress period and saves only the

net recharge (sometimes called the well term) in a file named E110712A.net. This run requires about 72MB of memory and takes about 10 minutes to create the input files.

The input files are typically named with the same file root and differing extensions. An arbitrary model input file name can be specified using the **- -ext <filename>** flag.

A number of files are used to specify quantities that apply to all periods of the simulation. These files are read once when **mkmod** starts. The *.mdl file is used to specify the model units, dimensions, stress period and similar data. The *.cel file specifies the active cells and cell areas. The *.sol file specifies the soil type for non-irrigated acreage. The *.red file specifies the reduction factors for gravity and sprinkler irrigation. The *.eff file specifies On-Farm algorithm parameters such as the efficiency, DPin and DPex by entity.

Most of the model inputs specify a header which defines the various features and then a set of data for each stress period. The *.ent file defines the entities ET adjustment factors, and the sprinkler fraction by entity for each stress period. The *.iar file specifies the irrigated acres by entity. The *.div file specifies the diversions and returns for each entity. The *.cnl file specifies the canal leakage by entity. The *.fpt and *.off files specify the fixed point and off-site pumping by entity, respectively. The *.pch file specifies the perch river seepage by reach. The *.trb file specifies the tributary underflow by reach.

There are three inputs that are specified for every cell and stress period. The *.pre, *.eti and *.nir files contain the arrays of precipitation, evapotranspiration and non-irrigated recharge, respectively. The input files are described in design documents that can be found on the web at http://www.idwr.idaho.gov/Browse/WaterInfo/ESPAM/ESPAM_2_Design_Docs/.

mkmod parameters

The **mkmod** program is typically run as

mkmod <fileroot>

In this minimal form, the input and output names are named by appending extensions to the specified file root. Optional parameters may be specified before the file root.

Inputs default to the file root with an extension. However, different names for individual input files may be specified by using the extension name. So, for example, if the files are named E110712A but the model file should be *foo.mdl,* the program can be run as

mkmod - -mdl foo.mdl E110712A.

Note that there are two hyphens before the extension, to distinguish the multiple letter extension names from single letter parameters.

The output files will use the same file root as the input files. However, a different file root can be set using the **-o** parameter.

The **-a** parameter will cause **mkmod** to save the service areas for each entity to a file named by the entity name and a .dat extension.

By default, **mkmod** will save each modeled stress to a different output file with the steady state period as the first stress period. Calculating the steady state initial stress period requires **mkmod** to maintain all stress periods in memory before writing the output files. This requires a lot of memory. If memory is limited, **mkmod** can be run without the steady state initial period. The following parameters control the output of **mkmod**:

-s Output a single output file (net recharge or "well term" only).

-ss Output a single output file without steady state.

-sss Output only the single output file without steady state and no other tables.

-S No steady state, but separate stress files.

The **-sss** option is intended for use during the PEST calibration runs where no additional output is desirable.

The **-m** parameter is used to select which algorithm should be used to calculate recharge and pumping. The **-m** parameter must be followed by one of the key words:

FRS On-Farm algorithm with soil moisture and calculated runoff used as returns.

FER On-Farm algorithm using calculated runoff for returns, but no soil moisture calculations.

MFE On-Farm algorithm, returns read from DIV file, no soil moisture calculations.

1 ESPAM 1.1 algorithm

mkmod output files

Files written by **mkmod** use the same file root as input files, or a distinct file root if the **-o** option is specified.

The net recharge or "well term" file uses the **net** extension and is stored in the MODFLOW well file format. Each stress period starts with the number of cells saved followed by the string STRESS PERIOD and the stress period number. This is followed by an entry for every cell with a nonzero stress. Note that the file uses free format for these fields.

If instead separate output files are requested (-S option), the following output files are generated:

ppt precipitation recharge;

can canal seepage;

tri tributary underflow;

gwr groundwater deep percolation;

swr surface-water deep percolation;

wel well pumping;

The wel file is in the MODFLOW well file format. The other files are in MODFLOW recharge file format. Some versions of MODFLOW allow multiple recharge and similar stress terms to be specified, which makes tracking the effect of different budget terms throughout the process simpler.

A summary output table is saved with the **htm** extension. This table summarizes stresses by individual entities for all stress periods. In addition summaries for various groups of entries are provided. A summary table with the **dat** extension is also produced. The columns in this table are labeled in the **htm** table, however, the **dat** file is better suited for graphing and similar post-processing.

Return flows calculated by **mkmod** are saved using the **rfl** extension. Since only surface water entities can generate return flows, only surface water entities appear in this file.

Acreages on a cell by cell basis can also be saved by **mkmod** for each stress period. The extensions used are:

AGWgr groundwater irrigated gravity acres;

AGWsp groundwater irrigated sprinkler acres;

ASWgr surface-water irrigated gravity acres;

ASWsp surface-water irrigated sprinkler acres;

Additional debuging will be produced as requested in the *.mdl file. The file extensions are specified in the *.mdl file.

Programming notes

The **mkmod** program is written as a series of functions to mostly operate on a small number of global variables. The **mkmod** program uses period totals for all variables, typically volumes. The recharge and pumping calculations are done on a per unit area basis, and translated to volumes using the acreage.

Most inputs are stored in the hash **x**. Only one stress period at a time is read, so **x** only contains values for the current period. The primary index is the type of the data, using the file extension as the index.

Outputs are stored in the hash **out**. The values are indexed as **out{type}[period][cell]** where the type corresponds to the extension of the output file name. The periods are numbered from one. Period zero would be the steady state, if there is one. Cells are a linear index. All values are volumes.

Summary outputs by entity are saved in the array **sum** and are indexed as **sum[period]{entity}{type}**. The values are typically volumes, but in a few instances are values such as efficiencies.

The following lists the major functions used:

The **Read** function reads the next data line from the file. Lines beginning with a # sign are considered to be comments and are skipped. Blank lines are also skipped. It returns the line read. The function will return ***undef*** if an end of file is encountered, unless an optional second parameter is specified, in which case the text of the second parameter is printed and the program will terminate.

The **ReadLine** function will call **Read** to read the next data line, and then split the line on spaces. It will return the fields read as an array, or an empty array on end of file.

The **ReadCSV** funcion will read a file of comma separated values that should provide one value for every cell in the domain. This function is used to read the soil and ibound arrays.

The **Open** function is used to open input data files and read the header. The type parameter is used to define whether the file contains entities, point, line or array data. The header and a file handle are returned as a hash that will be used by the **Next** function.

The **Next** function is used to read the data for the next stress period. This function uses the header information returned by the **Open** function to determine how many records should be read, and maps the data to specific entities, reaches or cells, depending on the type of data read.

The **Average** function is used to calculate the average rates for the summary tables.

The **SaveCell** and **SaveArea** functions are used to save stresses in MODFLOW well (cell) or recharge (array) format respectively. MODFLOW requires quantities to be stored as rates, so these functions will convert the volumes used by **mkmod** into rates before writing the files.

Brannon Review

The following description of **mkmod** was prepared by Jim Brannon and is included here with minor formatting changes.

MKMOD 5 "On-Farm Water Budget" Code Review

by Jim Brannon

Introduction

During 2010 the IDWR ESPAM2.0 model development included recommendations by ESHMC committee members to add more complex "On-Farm water budget algorithms" to the MKMOD tool code. These algorithms were presented and discussed (led by Greg Sullivan) at several ESHMC meetings. Dr. Willem Schreuder implemented these algorithms into the MKMOD PERL code and ran tests to verify they were working.

However, due to the increased complexity of the new algorithms and how critical they are to the ESPAM 2.0 results, the ESHMC wanted an independent verification that the code correctly implemented the exact algorithms that were designed, discussed and approved in the meetings. PERL code is dissimilar to the more familiar engineering programming languages like FORTRAN and Visual Basic, so other ESHMC members were unsure of their ability to readily understand the code. Having significant programming experience in several languages (though not PERL) Jim Brannon (Leonard Rice Engineers) offered to attempt to review the code on behalf of Rangen for the ESHMC.

Description

Dr. Schreuder provided the latest version of the PERL MKMOD code, both MKMOD4 and MKMOD5. The version changed from 4 to 5 during the review. After a short, intense tutorial session on the basics of PERL, Jim Brannon flow charted the larger code structure in general until the sections covering the On-Farm water budget could be confidently identified. These code sections were then flow charted and studied in detail.

Running multiple data sets through the code was outside the scope of the effort, so in order to verify the code, the code logic was carefully converted into diagrams that represent explicitly each logic case encountered. These diagrams were converted into electronic documents made available to IDWR and the ESHMC members.

Questions and comments by the reviewer were inserted into a copy of the PERL code, also made electronic and available to the IDWR and ESHMC members.

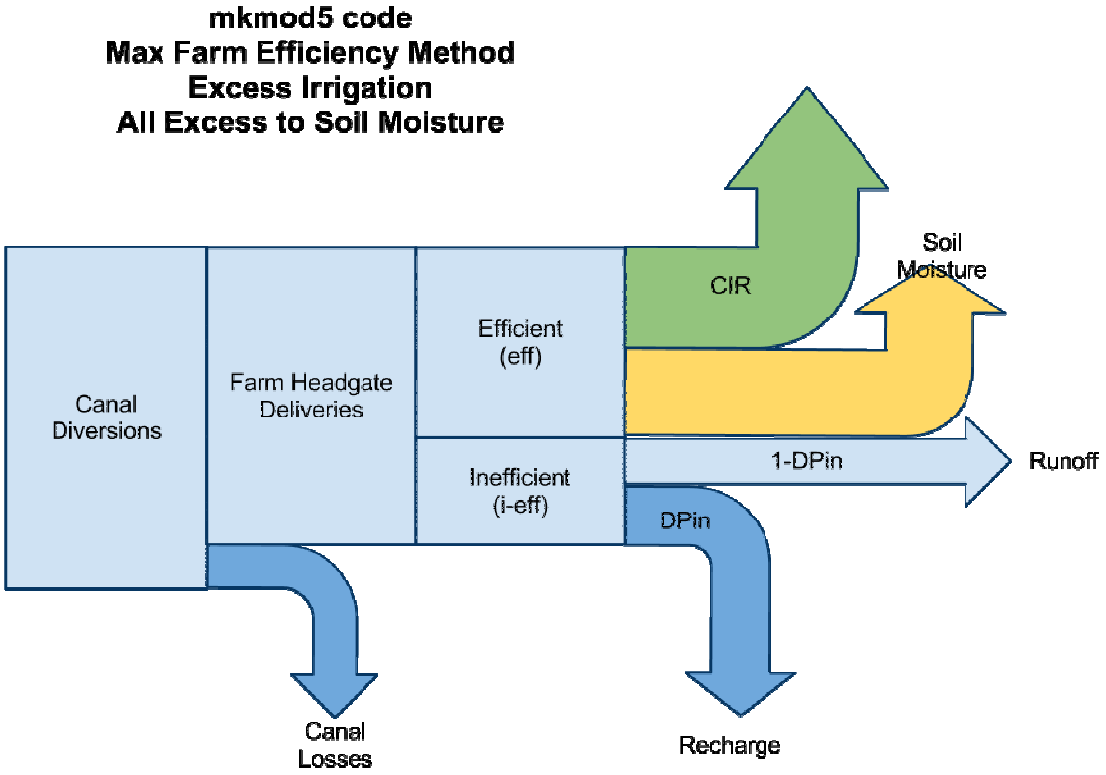
Results

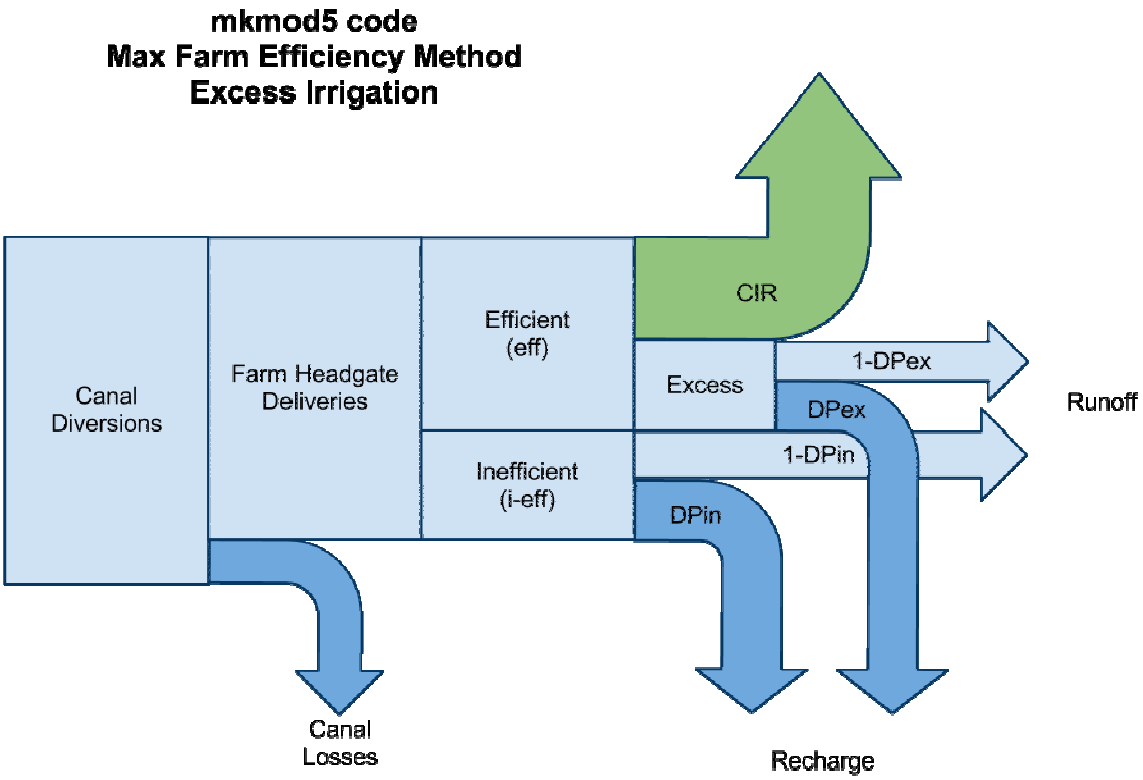
To the reviewer, Jim Brannon, each code case appeared consistent with the algorithms as described in the ESHMC meetings. The operations in some parts of the code were uncertain during the review and noted in the code, but further discussions with Dr. Schreuder cleared them up. These code logic

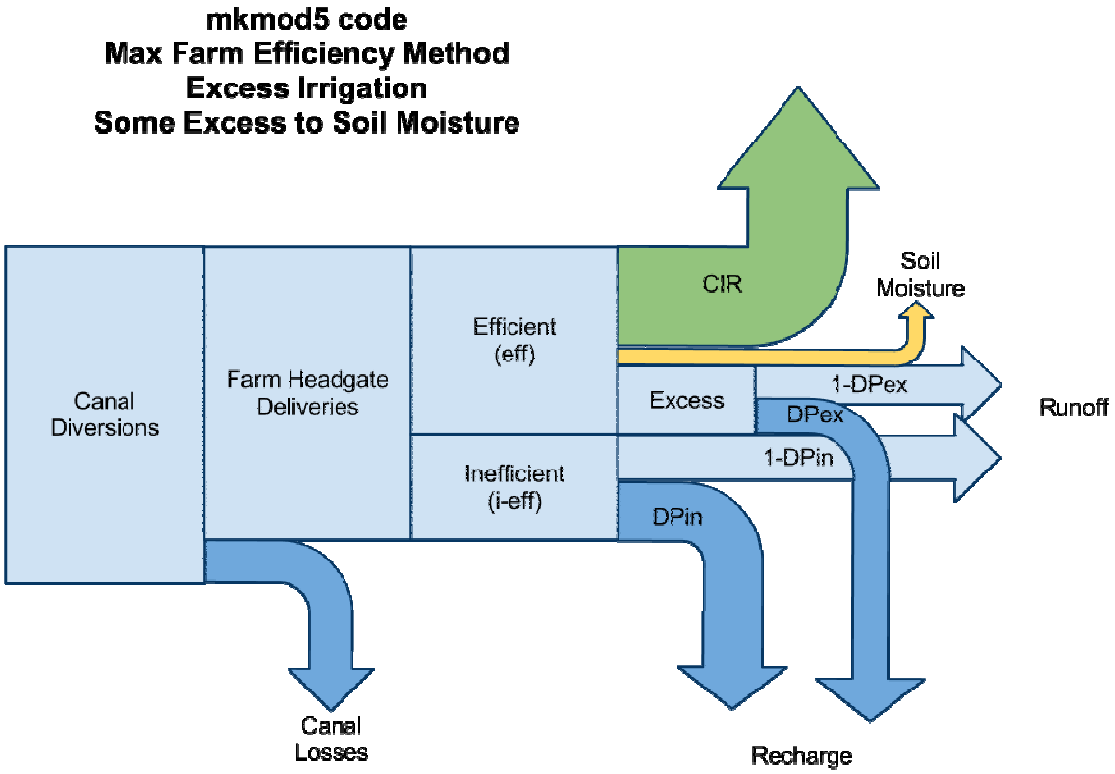
diagrams were then presented to the ESHMC in November 2010, to get the whole committee's comments and approval. During the meeting the ESHMC agreed that they represented the algorithms that had been designed and discussed during prior meetings.

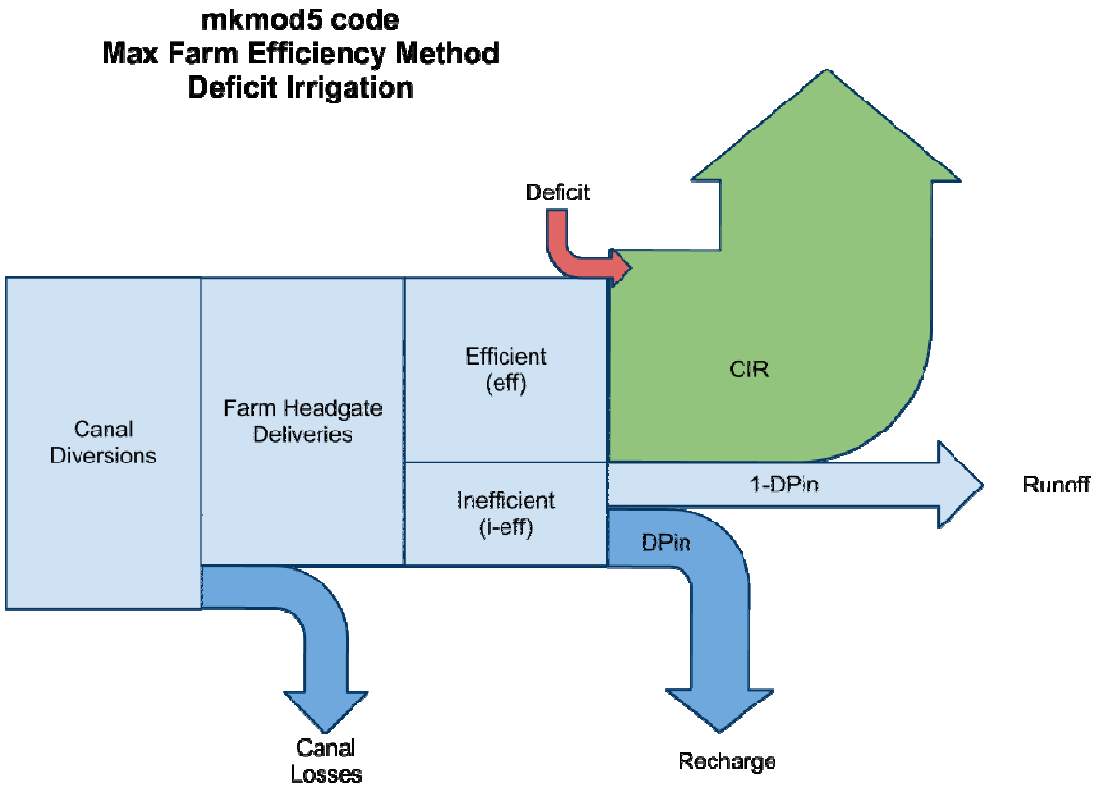
The code has been used in MKMOD5 through MKMOD8, though no additional verification of the code has been done by Jim Brannon since November 2010 (MKMOD5).

Below are the diagrams of the logic cases encountered in the code and presented to the ESHMC.

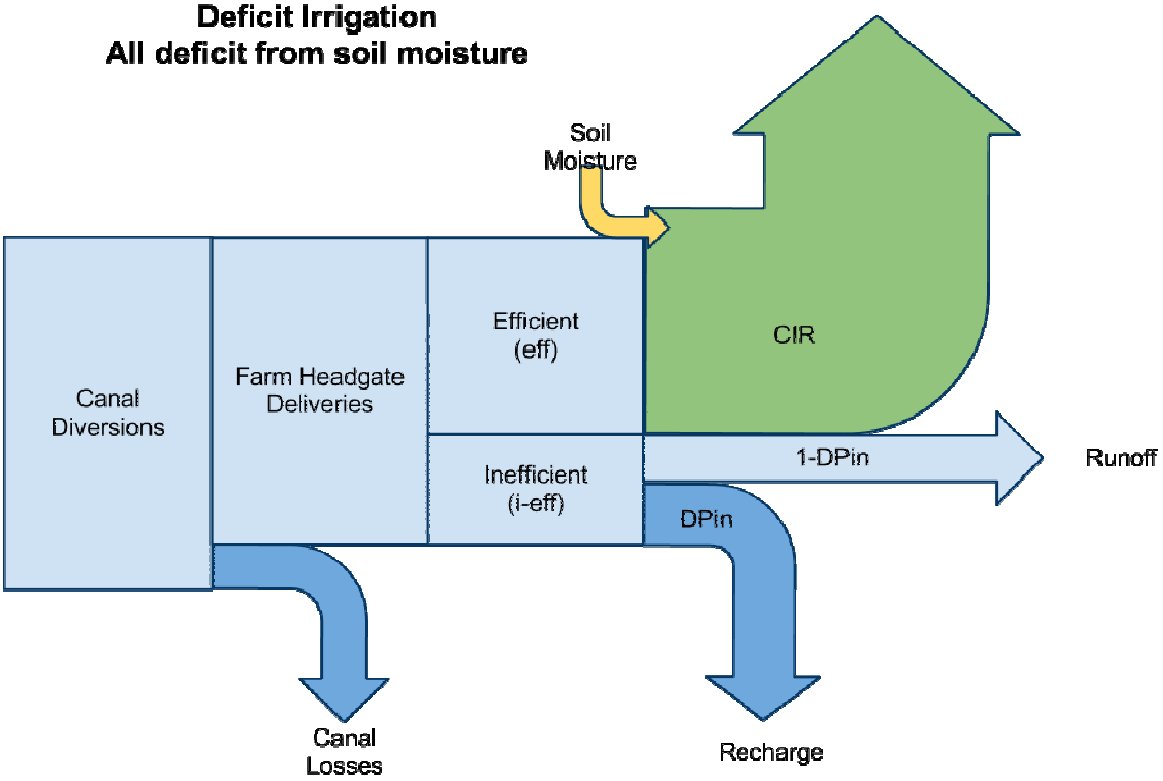

mkmod5 code
Max Farm Efficiency Method
Excess Irrigation
All Excess to Soil Moisture
Canal Diversions
Farm Headgate Deliveries
Efficient (eff)
Inefficient (i-eff)
CIR
Soil Moisture
1-DPin
Runoff
DPin
Canal Losses
Recharge


mkmod5 code
Max Farm Efficiency Method
Excess Irrigation
Canal Diversions
Farm Headgate Deliveries
Efficient (eff)
Inefficient (i-eff)
CIR
Excess
1-DPex
DPex
1-DPin
DPin
Runoff
Canal Losses
Recharge

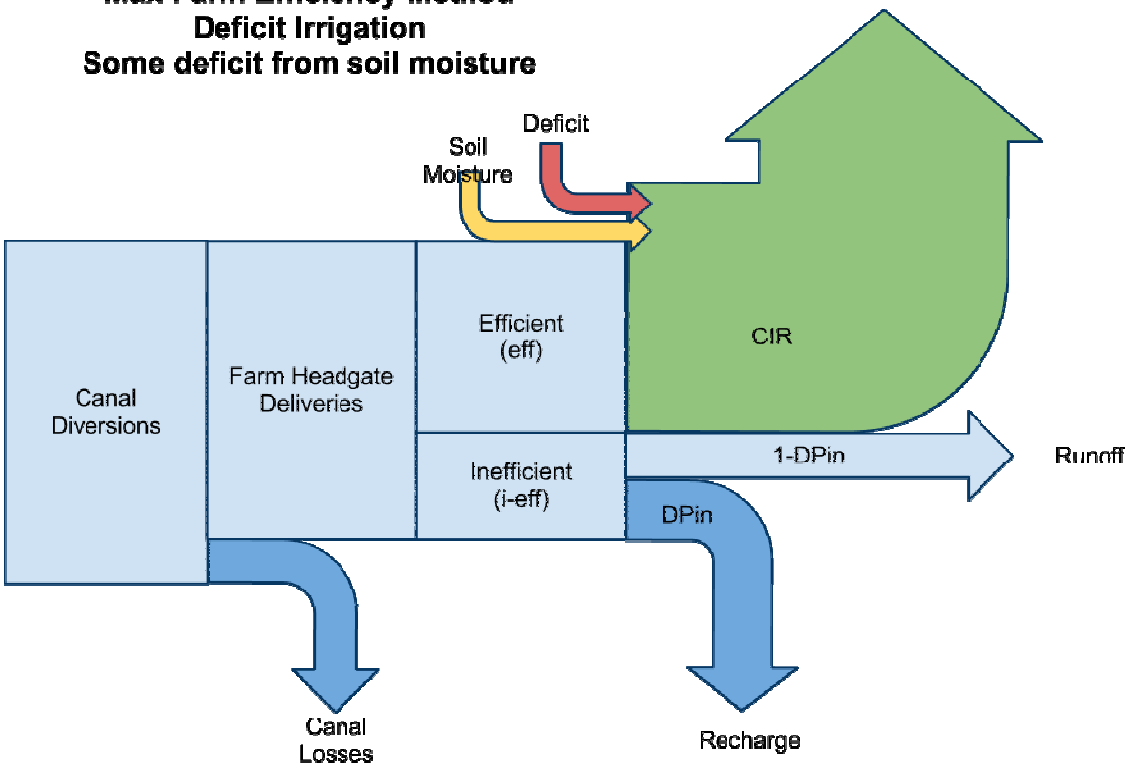

mkmod5 code
Max Farm Efficiency Method
Excess Irrigation
Some Excess to Soil Moisture
Canal Diversions
Farm Headgate Deliveries
Efficient (eff)
Inefficient (i-eff)
CIR
Soil Moisture
Excess
1-DPex
DPex
1-DPin
DPin
Runoff
Canal Losses
Recharge


mkmod5 code
Max Farm Efficiency Method
Deficit Irrigation
Deficit
CIR
Canal Diversions
Farm Headgate Deliveries
Efficient (eff)
Inefficient (i-eff)
1-DPin
Runoff
DPin
Canal Losses
Recharge

mkmod5 code
Max Farm Efficiency Method
Deficit Irrigation
All deficit from soil moisture



mkmod5 code
Max Farm Efficiency Method
Deficit Irrigation
Some deficit from soil moisture



Britton Review

The following outline of **mkmod** was prepared by Ben Britton. A few instances of **mknod** have been converted to **mkmod**.

MKMOD v8.1

This is commentary on ***mkmod81.pl***. A presentation containing the description of ***mkmod***, its command-line switches, input/output files, programming notes and flowchart/algorithms is available at http://www.prinmath.com/eshmc/mkmod8/mkmod.pdf

The program is very well documented. As the header states, it creates MODFLOW input files for ESPAM. It tracks total and net stresses. The numbers below refer to line numbers in ***mkmod81.pl***.

1-52
Explanation of input-file types.

54-61
Set the allowable file-extensions used for input to MKMOD81 as:
'mdl','cel','sol','red','ent','fpt','off','div','cnl','trb','pch','pre','eti','nir','iar','eff'
Uses GetOpt to read the command-line options the user supplied.
Get the root input file-name and stores it in $INPUT. If not supplied, the program ends.
Get the root output file-name, if specified with the –o switch, and stores it in $OUTPUT.
Set the default for command-line switches (o,m, s, S and a) and then reads any switches from the input.

64-69
Determine ***Surface water method (switch -m)***
Defaults to FRS (""Maximum Farm Efficiency using Runoff for Returns and Soil Moisture"") if not specified as one of the following:
FER – "Maximum Farm Efficiency using Runoff for Returns",
FRS – "Maximum Farm Efficiency using Runoff for Returns and Soil Moisture");

71-89
Set the arrays that make up net recharge, including conversion factors and output-file extensions.

91-98
Set default input file names

100-136
Define global variables and subroutines, ***Read*** and ***ReadLine***, to read from the input files.

138-204
Open the **.mdl** file and read the following:

Title1, Title2, time unit, length unit, number of stress periods and steady state definition, stress period lengths and descriptions, number of model rows/columns/layers, groundwater pumping layer, adjustment factors, etc.

206-218
Set the array of periods used for steady state.

220-233
Set a variable to represent a square mile, in feet. Calculate the total number of model cells. Open the **.cel** file and read CSV formatted data – one value for each row/column/layer.

235-267
Define the ***ReadCSV*** subroutine, which reads CSV-formatted data.

268-386
Define the ***Open*** subroutine, which opens and reads the header from the specified input file.

388-498
Define the ***Next*** subroutine, which reads the "next" stress period from the specified input file. It populates an array named **DATA** for that period. This subroutine uses ***ReadCSV*** and ***ReadLine***.

503-532
Open the **.sol** file and reads the soil types.

533-546
Open the **.red** file and read the reduction factors for gravity and sprinkler, adjust those factors.

548-560
Open the **.eff** file and read the irrigation efficiency parameters and soil properties.

562-575
Open the **.ent** file and read the irrigation entities. Check data and set up entities pointer.

577-588
Open the **.fpt** file and read the fixed point diversions.
Open the **.off** file and read the off-site groundwater pumping wells.
Open the **.div** file and read the canal diversions and returns.
Check that off-site wells are assigned to legal surface water entities.

590-595
Open the **.cnl**, **.trb** and **.riv** files and read the canals, tributaries and perched streams.

605-631
Open the output files, using the root output file-name specified on the command line.

633-675
Define a ***Save*** subroutine, which writes to an output file.

677-1082
Process data for each transient stress period.
  # Read data for this stress period
  # Check canal seepage rates in [0,1]
  # Check that precip and irrigated ET are non-negative
  # Non-irrigated recharge may be negative
  # Initialize summaries for this stress period
    # Applied, recharge, consumptive use (ET), area
    # For surface water entities, also diversion, seepage, returns, deficit irrigation and soil moisture
  # Sum total irrigated area by entity
  # Sum sprinkler and gravity areas by entity by cell
  # Sum non-irrigated areas by cell
  # Loop over cells in irrigated area list
      # Adjusted sprinkler and gravity area
      # Remove irrigated areas from non-irrigated array
      # Add irrigated area to this entity's irrigated area array
      # Add irrigated area to this entity's total area
      # Add irrigated area to list of cells irrigated by this entity
      # Add irrigated area to output array of irrigated area

  # Calculate total diversion, applied volume and seepage by surface irrigation entity
  #   sum holds totals by entity
  #   DIV diverions (including off-site pumping)
  #   APP applied water (including off-site pumping, excluding seepage)
  #   SEEP canal seepage
  #
  # Applied = Diversion - Returns
  # Add off-site pumping (Q<0)
  # Subtract canal seepage

  # Calculate application rate for surface water entities
    # Default value (Groundwater or error)
    # Surface water only (skip others)
    # Check application rate and irrigated acres
    # Application rate = Volume / Area

  # Calculate and distribute applied water and precipitation on irrigated lands
  # Loop over all irrigation entities
    # Distribute to sprinkler and gravity lands
Crop Irrigation Requirement
        # Groundwater irrigation
        # Net Recharge = Precip - Adjusted ET = -CIR = Recharge - Pumping
        # Pumping = CIR / Irrigation Efficiency
        # Recharge = Precip + (1-Irrigation Efficiency)*Pumping

          # Groundwater recharge for the cell is the rate time area
          # Groundwater pumping for the cell is the rate time area
          # Add pumping to applied water total for entity

```
        # Surface water irrigation using Maximum Farm Efficiency method
        # Net Recharge = Precip + Application - Adjusted ET = Application - CIR
          # Calculate excess irrigation or deficit
            # Initialize soil moisture to field capacity on first occurrence
            # Soil moisture source
            # Soil moisture sink
            # Deficit irrigation may take water from soil moisture
            # Irrigation excess may sink water to soil moisture
            # Adjust soil moisture array
          # Recharge is the sum of initial and excess deep percolation
          # Runoff is the sum of initial and excess surface runoff
          # Surface water recharge for the cell is the rate time area
        # Surface water irrigation using ESPAM 1.1 method
        # Net Recharge = Precip + Application - Adjusted ET = Application - CIR
          # Surface water recharge for the cell is the rate time area
        # Accumulate totals for the entity

# Precipitation on non-irrigated lands
# Recharge = Soil Type Adjustment * Rate * Area
  # Skip dead cells and cells fully irrigated
  # Adjust precipitation rate for soil type
  # Precipitation recharge is rate time non-irrigated area
  # Accumulate total precipitation recharge and non-irrigated area

# Fixed point and off-site adjustments
    # Array to adjust
    # Magnitude of adjustment (WetAdj is a global and is currently 1 so this does nothing)
    # Accumulate adjustment to output array
    # Accumulate magnitude of the adjustment

# Distribute canal seepage
# Amount is calculated based on diversion above
# Loop over canals
  # Distribute uniformly over cells
  # Add rate to cells

# Distribute tributary underflow and perched river recharge
    # Distribute uniformly over cells and apply scale factor
    # Add rate to cells

# Calculate application rate for groundwater water entities
  # Groundwater water only - skip others
  # Check irrigated acres
  # Application rate = Volume / Area

# Summarize results by GW/SW
  # Surface water only variables
  # Sum sprinkler area
```

Compute sprinkler fraction by type
Compute average application rate

Calculate irrigation efficiency (CIR/Applied)

Calculate net recharge
Net = Non-irrigated precip + Surface Recharge + Groundwater Recharge + Canal Leakage + Tributaries - Pumping

If immediate (no steady state) save to file an forget all arrays

If single stress out forget the other arrays to save memory

1084-1177
Calculate steady state totals (volumes) for each cell. Save results to output files.

1180-1195
Save service areas to file if requested (-a flag).

1197-1203
Define the ***date*** subroutine, which returns the hypertext markup for the current period.

1206-1240
Define the ***data*** subroutine, which returns the data for period and entity – SW, GW, non-irrigated areas, etc.

1243-1285
Define the ***Average*** subroutine, which calculates average rates for summary tables.

1287-end
Define text for output-files, then write output files.

Input files with data that apply to all periods in the simulation

.mdl Model units, dimensions, periods, ...
.cel Active cells and cell areas
.sol Soil type for non-irrigated recharge
.red Reduction factors (Gravity,Sprinkler)
.eff On-Farm parameters Eff,DPin,DPex,..

Input files with data by Entity

(Header + data set for each stress period)
.ent Entity names; sprinkler fractions
.iar Irrigated acres by cell
.div Diversions and returns

.cnl Canal leakage
.fpt Fixed point pumping
.off Off-site pumping
.pch Perch river seepage
.trb Tributary underflow

Input files with cell arrays

(no header, just array of values for each stress period)
.pre Precipitation
.eti Evapotranspiration
.nir Non-irrigated recharge

Output files

Net recharge ['well term'] (.net)
– MODFLOW well file format (k,i,j,q)
Summary table (.htm)
– Summarizes input and output by groups and
by entity by stress period
Summary table (.dat)
– Main summary table for plotting
Return flows (.rfl)
Debug output (.rfx)

Output files (separate terms)

.ppt precipitation recharge
.cnl canal seepage
.tri tributary underflow
.gwr groundwater deep percolation
.swr surface water deep percolation
.wel well pumping

Output Files (acreage)

.AGWgr Groundwater Gravity Acres
.AGWsp Groundwater Sprinkler Acres
.ASWgr Surface Water Gravity Acres
.ASWsp Surface Water Sprinkler Acres

Comparison with IDWR hand calculations

The Schreüder overview, and Brannon and Britton reviews outline what MKMOD is supposed to do, but does it perform as expected? Tables 1 and 2 below contain output from MKMOD and hand calculations performed by Allan Wylie (IDWR) respectively. The columns in Table 2 with bold numbers can be compared with MKMOD output in Table 1.

Table 2 follows the flow diagram in Brannon's review labeled "mkmod5 code Max Farm Efficiency Method Excess Irrigation." Brannon's diagram indicates that canal losses plus farm headgate deliveries should equal diversions. The second column in Table 2 labeled **diversions** is the sum of canal losses and farm headgate deliveries and can be compared with the column labeled **Diverted** in Table 1.

The next ten columns in Table 2 contain calculations necessary to make more comparisons. Brannon's diagram shows that farm deliveries are broken into efficient and inefficient portions. As noted above, sprinkler irrigation is assumed to have a maximum efficiency of 0.85 and gravity irrigation is assumed to have a maximum efficiency of 0.80. The column labeled **sp frac** in Table 2 is the fraction of farm deliveries delivered to sprinkler irrigated lands and the column labeled **g frac** is the portion delivered to gravity irrigated lands. The columns labeled **sp inef** and **g inef** are computed by multiplying the fraction of farm deliveries for sprinkler irrigated land and gravity irrigated land by the quantity (1-efficiency).

Both Table 1 and Table 2 contain columns labeled **Excess**. This represents water delivered to the farm headgate that is beyond what is necessary to supply the crop irrigation requirement at maximum irrigation efficiency. The Brannon diagram indicates that this is computed by subtracting the crop irrigation requirement from the efficient portion of the farm headgate deliveries. The columns labeled **Excess** can be directly compared, and the differences are small, likely the result of rounding errors.

The Brannon diagram indicates that the inefficient and excess fractions are then divided between runoff and recharge. Recharge from the inefficient fraction is computed by multiplying the sprinkler and gravity inefficient fractions by DPin. The results of these calculations are displayed in Table 2 in the columns labeled **sp rchg** and **g rchg**. Recharge from the excess fraction is computed by multiplying the excess fraction by DPex. The result of this calculation is displayed in Table 2 in the column labeled **ex rch**. Total recharge is then the sum of the inefficient recharge from the sprinkler and gravity fractions and from excess. The result of this calculation is displayed in Table 2 in the column labeled **Tot rch** and this can be directly compared with the MKMOD output in Table 1 in the column labeled **Recharge**. Again, the differences are small, perhaps the result of rounding errors.

Total runoff is the sum of runoff from the inefficient and excess fractions. Runoff from the inefficient fraction is computed by multiplying the sprinkler and gravity fractions by (1-DPin). The results of these calculations are displayed in Table 2 in the columns labeled **sp roff** and **g roff**. Runoff from the excess fraction is computed by multiplying the excess fraction by (1-DPex). The result of this calculation is displayed in Table 2 in the column labeled **ex roff**. Total runoff is then the sum of the runoff from the inefficient and excess fractions. The result of this calculation is displayed in Table 2 in the column labeled **Tot roff** and this can be directly compared with the MKMOD output in the column labeled **Runoff** in Table 1. The differences between the columns in these tables are likely due to rounding errors.

This analysis shows that MKMOD is conducting the calculations in the On-Farm Algorithm correctly.

Table 1. MKMOD output for selected surface water entities.

MKMOD Output										Parameters	
Enitiy	Name	Diverted (af)	Canal Seepage (af)	Farm Delivery (af)	CIR (af)	Sprinkler (%)	Runoff (af)	Recharge (af)	Excess (af)	DPin	DPex
IESW000	Null	140575	0	140575	29408	55.9	0	111167	86979	1.00	1.00
IESW011	ButteMrk	88617	13293	75324	40517	56	696	34110	21851	0.98	0.98
IESW012	Canyon	27911	2233	25678	3622	88.9	0	22055	18058	1.00	1.00
IESW018	Falls	24200	2419	21780	341	100	0	21439	18172	1.00	1.00
IESW027	Milner	59048	23619	35429	8637	27.8	969	25821	20203	0.97	0.96
IESW034	Peoples	290116	121831	168284	49036	74.5	31747	87497	91867	0.74	0.73
IESW038	Rexburg	54171	22754	31417	8204	24.9	8878	14333	17320	0.62	0.62
IESW039	Chester	18865	5660	13205	1946	26.9	4318	6940	8795	0.61	0.62
IESW044	Montview	100428	20083	80344	50697	23.9	0	29647	14546	1.00	1.00
IESW052	Small	13086	0	13086	3205	3.4	0	9881	7285	1.00	1.00
IESW055	Labelle	267251	82843	184408	51940	4.7	41757	90709	96026	0.71	0.67
IESW058	AmFalls2	146475	112786	33689	17648	24.2	0	16041	9709	1.00	1.00

Table 2. Hand calculations used to check MKMOD.

Hand Calculations																
Name	**diversions**	**sp frac**	**sp eff**	**sp inef**	**sp rchg**	**sp roff**	**g frac**	**g eff**	**g inef**	**g rchg**	**g roff**	**Excess**	**ex rch**	**ex roff**	**Tot rch**	**Tot roff**
	cnl+deliv	**sp% * deliv**	**sp frac * .85**	**sp frac * .15**	**dpin*sp inef**	**(1-dpin) * sp inef**	**(1-sp%) * deliv**	**g frac * .80**	**g frac * .20**	**dpin * g ineff**	**(1-dpin) * g inef**	**(sp eff + g eff) - cir**	**(dpex * excess)**	**(1-dpex) * excess**	**sp rchg + g rchg + ex rchg**	**sp rof + g rof + ex rof**
Null	**140575**	78581	66794	11787	11787	0	61994	49595	12399	12399	0	**86981**	86981	0	**111167**	**0**
ButteMrk	**88617**	42181	35854	6327.2	6200.7	126.54	33143	26514	6629	6496	132.57	**21851**	21414	437.03	**34111**	**696**
Canyon	**27911**	22828	19404	3424.2	3424.2	0	2850.3	2280	570.1	570.1	0	**18062**	18062	0	**22056**	**0**
Falls	**24199**	21780	18513	3267	3267	0	0	0	0	0	0	**18172**	18172	0	**21439**	**0**
Milner	**59048**	9849	8372	1477.4	1427.6	49.79	25580	20464	5116	4944	172.41	**20199**	19451	747.69	**25822**	**970**
Peoples	**290115**	1E+05	1E+05	18806	13853	4952.3	42912	34330	8582	6322	2260.1	**91860**	67324	24536	**87500**	**31748**
Rexburg	**54171**	7823	6649	1173.4	725.07	448.35	23594	18875	4719	2916	1803	**17321**	10693	6627.42	**14334**	**8879**
Chester	**18865**	3552	3019	532.82	327.38	205.44	9652.9	7722	1931	1186	744.36	**8796**	5427.1	3368.51	**6941**	**4318**
Montview	**100427**	19202	16322	2880.3	2880.3	0	61142	48913	12228	12228	0	**14538**	14538	0	**29647**	**0**
Small	**13086**	444.9	378.2	66.739	66.739	0	12641	10113	2528	2528	0	**7286**	7286	0	**9881**	**0**
Labelle	**267251**	8667	7367	1300.1	928.42	371.65	175741	1E+05	35148	25100	10048	**96020**	64682	31338.2	**90710**	**41758**
AmFalls2	**146475**	8153	6930	1222.9	1222.9	0	25536	20429	5107	5107	0	**9711**	9710.8	0	**16041**	**0**

Attachment A: Reviewer Comments and IDWR responses

November 28, 2012 Comments from Bryce Contor

Dear Rick -

The version of Appendix B which I have is a *.pdf, so I have not provided "track changes" edits. Here are a few comments:

1) Page 3. The introduction should more explicitly point out that the On Farm algorithm is a novel approach developed especially for ESPAM2.x.

Accept

2) Pages 9, 10, 17. There are references to MKMOD4, MKMOD5, MKMOD8 and MKMOD8.1. The appendix should explicitly state which iteration of MKMOD was used to generate the water budget from which ESPAM2.1 was calibrated. It should provide a summary of the differences between that version and the versions described in the appendix. Likely only Dr. Schreuder would be able to provide that description, though it would be best if it could be an independent explanation.

Accept, analysis and text provided by IDWR.

3) Page 17 header. Perhaps "MKNOD" was meant to be "MKMOD" :)

Accept

4) Page 23. Dr. Schreuder asserts that MKMOD can be run in V1.1 mode to duplicate the results of the ESPAM1.1 calculation algorithms. ESPAM1.1 has the ability to impute groundwater pumping to satisfy ET if needed, regardless of the nominal water source in the data. I am not convinced that MKMOD includes this functionality, even when run in V1.1 mode.

I request that the coding of the MKMOD V1.1 option be carefully and independently reviewed, and that Appendix B explicitly assert the preservation or acknowledge the omission of this functionality. I realize that some consider this functionality to be a flaw, but in any case the documentation should clearly indicate its presence or absence in MKMOD mode V1.1.

Accept, a reference to an analysis conducted in April of 2009 was inserted into the Introduction.

5) Page 23. Either the introduction or this section of the description should explain the important philosophical difference between the methods: ESPAM1.1 asserts that the ET estimates are more reliable than the water-source designations and canal seepage estimates; MKMOD makes the opposite assertion.

The results of this difference are:

* The ESPAM1.1 algorithm over-estimates mixed-source withdrawals and under-estimates recharge, if ET estimates are too high relative to surface-water diversion estimates. MKMOD has robustness through the initial loss fraction and the DPin parameter.

* ESPAM1.1 is robust to incorrect assignment of water source; its only response is local distortion in spatial distribution of recharge. Incorrect assignment of water source can cause MKMOD to over-estimate recharge and under-estimate mixed-source withdrawals.

* ESPAM1.1 is robust to imprecision in canal seepage; the only result is local distortion in spatial distribution of recharge. The MKMOD response to too-high canal seepage can be over-estimation of recharge and under-estimation of mixed-source withdrawals.

* With both algorithms, too-low canal seepage results only in local distortion in spatial distribution of recharge.

* In either case, recharge will be over-estimated and mixed-source extraction under-estimated if ET estimates are too low relative to surface-water diversions.

* Imprecision in return-flow estimates affect both methods. Returns estimates are applied directly in ESPAM1.1 algorithms and are implicit in the DPin and DPex parameters in MKMOD. Under-estimating returns causes an over-estimate of recharge, while over-estimating returns causes an under-estimate of recharge.

Accept, these bulleted items are added to the introduction

6) Either the introduction or the Page 23 description should point out that in cases of surface-water shortage in the absence of supplemental wells, the ESPAM1.1 algorithm requires manual adjustment to avoid under-estimation of recharge. MKMOD automatically makes an adjustment, to the extent allowed by the initial loss fraction and DPin parameters.

Accept, this is added in with the above bulleted items.

7) Except as noted in comments 4 through 6, I did not proofread the equations and figures.

8) Except for comment 3 I did not proofread Mr. Britton's explication of the code.

Thanks for this opportunity to provide comments. This is not a suggestion that methodology be changed for ESPAM2.1, but only a request to clarify the documentation.

Bryce

--

Bryce A. Contor
Senior Hydrologist

482 Constitution, Idaho Falls, ID 83402

E-Mail: bcontor@rockymountainenvironmental.com
Alt. E-Mail: bcontor.rm@gmail.com
VOICE: 208-524-2353 ||| FAX: 208-524-1795 ||| CELL: 208-681-9100

January 12, 2013 B. Contor comments to Appendix B.

I appreciate inclusion of a comparison between the philosophy and capabilities of the ESPAM1.1 algorithms and the MKMOD algorithms. However, I believe the draft I provided in the last review is more accurate than has been presented in the version of Appendix B downloaded on January 12, 2013, in three specific cases:

A. Point two on page three should more correctly represent that ESPAM1.1 algorithms result in errors of spatial distribution in such cases but *do not bias the water budget,* whereas MKMOD algorithms result in both spatial imprecision *and bias.*

Accept.

B. Point three on page four should more clearly point out that the result of the ESPAM1.1 algorithm is that the net water budget honors the data and is unchanged by the imprecision in apportioning diversion to seepage, while the MKMOD algorithm is unable to preserve the water budget under similar errors in canal seepage. That is, ESPAM1.1 is able to offset too-high seepage with too-high or even fictitious pumping, where MKMOD has no automatic, internal mechanism.

Accept.

C. Point four on page four would be more accurately worded "MKMOD parameters can be set so that the appropriate reduction in consumptive use occurs automatically;" in this case MKMOD is more powerful than point four seems to indicate.

Accept.

Additionally, the paragraph that starts out "The shortcomings associated with..." properly applies to both algorithms; in either case, when problems are "identified, the sources of most mistakes can usually be repaired." The conceptual difference arises when problems are *not* identified:

A. MKMOD is robust to cases where ET is too high relative to SW diversions.

B. The algorithms are equally vulnerable when ET is too low relative to diversions.

C. ESPAM1.1 is robust to imprecision in representation of mixed source lands and groundwater contributions on mixed source lands, in terms of preserving the water budget. MKMOD will incorrectly reduce consumptive use if data imprecision triggers a false indication of deficit irrigation.

D. ESPAM1.1 is robust to overestimation of canal seepage, in terms of preserving the water budget; it has an ability to create an offsetting withdrawal to compensate for the excess recharge, and is not vulnerable to incorrectly reducing consumptive use due to a false indication of deficit irrigation.

E. The algorithms are equally able to preserve the water budget when canal seepage is too low.

Reject, we recognize that there are differences between the tools used for ESPAM1.1 and ESPAM2.1, however, this document is about mkmod, not about the ESPAM1.1 tools. It is easier for us to identify deficit irrigation, excess irrigation, higher or lower than expected ET, etc and map these by entity using the output files from mkmod than with the ESPAM1.1 tools.