Title:	**Water system research Yellow River Basin Improving agro-hydrological models**

Author:	J. Schellekens	Institute:	WL|Delft Hydraulics
Author:	C.J. Sprengers	Institute:	WL|Delft Hydraulics
Author:	P.J.A. Gijsbers	Institute:	WL|Delft Hydraulics

August 2003

Number of pages	:	25	
Keywords (3-5)	:	agro-hydrological modelling, irrigation, crop planning	
DC-Publication-number	:	DC1-626-1	
Institute Publication-number (optional)	:		
Report Type	:	☐	Intermediary report or study
	:	☒	Final projectreport
DUP-publication Type	:	☒	DUP Standard
		☐	DUP-Science

Acknowledgement
This research has been sponsored by DGIS (Ministry of Foreign Affairs) and the ICES-2 programme

Other Research project sponsor(s):

Abstract

The agricultural sector, and especially the irrigated agriculture, is one of the biggest sectors with respect to water withdrawal in a river basin. Therefore, proper planning of irrigation schemes and cropping patterns, and thus water demands, can be a useful method to improve the balance between water demand and water availability. Given the FAO methods to compute crop water requirements, the computation of water demands and a water balance is not a complicated task. However, the difficulty arises in two aspects, namely:

- the bio-physical variability over space and the feasibility to use raster-based models for both demand and allocation computations.
- proper administration of the areas being planted, the time window of soil preparation, seeding, rotation schedules for irrigation etc.

Within this project, a modelling exercise has been conducted using the PC-Raster software package to assess the combination of an on-line coupling between network based water allocation model of a river basin with a raster based crop water balance model for local agricultural schemes.
Based on the exercise, it was concluded that PC-Raster is a useful tool to develop a distributed hydrological model, but the software package is not fit yet for on-line linkage to a network based river basin allocation model such as RIBASIM. Linkage would become easier if PC-Raster becomes available as a library that can be used by other programming languages.

With respect to the planning and administration of cropping patterns, a software component has been designed under the name 'CropPlan'. After an analysis of potentially useful agro-hydrological model, it has been decided to incorporate the concepts, and some code, of the OMIS software package (Operation and Management of Irrigation Systems, a DOS-program) into the design. Major elements in this tool are:

- the locations of the irrigation schemes (connected to a map)
- the crops and their properties (parameters)
- the cropping pattern (i.e. cropping schedule)
- the hydrological conditions;
- the resulting local water balance
- linkages to other tools

PROJECT NAME:	**Water systeem onderzoek Yellow River Basin**	PROJECT CODE:	**06.02.06**
BASEPROJECT NAME:	**Water Systems**	BASEPROJECT CODE:	**06.02**
THEME NAME:	**Integrated Water Resources Managment**	THEME CODE:	**06**

Executive Summary

The agricultural sector, and especially the irrigated agriculture, is one of the biggest sectors with respect to water withdrawal in a river basin. Therefore, proper planning of irrigation schemes and cropping patterns, and thus water demands, can be a useful method to improve the balance between water demand and water availability. Given the FAO methods to compute crop water requirements, the computation of water demands and a water balance is not a complicated task. However, the difficulty arises in two aspects, namely:

- the bio-physical variability over space and the feasibility to use raster-based models for both demand and allocation computations.
- proper administration of the areas being planted, the time window of soil preparation, seeding, rotation schedules for irrigation etc.

Agro-hydrological modelling using a raster based concept

Objective of this modelling exercise was to assess the suitability of on-line linkage between a water allocation model for river basins (RIBASIM) and a raster based agro-hydrological models developed in PC-Raster. Within the network model, the raster based model is represented as one node.

Within water allocation models, an essential concept is the distinction between two phases within one time step, namely the demand or target setting phase and the actual allocation phase. Within the agro-hydrological model, the same distinction has been applied, using a simplified soil water reservoir as the balancing element. Note that in the test, the cells were not interconnected.

Based on the exercise, it was concluded that PC-Raster is a useful tool to develop a distributed hydrological model. With respect to the conceptual model design, the timing aspect (distinction between demand and allocation) is of major importance. At the moment, the desired variation in time stepping is not easy to implement. Therefore, the software tool is not yet fit yet for on-line linkage to a network based river basin model such as RIBASIM. Linkage would become easier if PC-Raster becomes available as a function based library (DLL) that can be called by other programming languages.

With respect to the planning and administration of cropping patterns, a software component has been designed under the name 'CropPlan'. After an analysis of potentially useful agro-hydrological model, it has been decided to incorporate the concepts, and some code, of the former OMIS software package into the design. Major elements in this tool are:

- the locations of the irrigation schemes (connected to a map)
- the crops and their properties (parameters)
- the cropping pattern (i.e. cropping schedule)
- the hydrological conditions;
- the resulting local water balance
- linkages to other tools

As the full design report was written in Dutch, this report contains only the essential elements of the design such as the data model, the relation to the database system of RIBASIM, the user interface, the computations and the control flow.

PROJECT NAAM:	**Water systeem onderzoek Yellow River Basin**	PROJECT CODE:	**06.02.06**
BASISPROJECT NAAM:	**Water Systems**	BASISPROJECT CODE:	**06.02**
THEMA NAAM:	**Integrated Water Resources Managment**	THEMA CODE:	**06**

Table of contents

1 Connecting raster-based agro-hydrological models to a network based water allocation model

Objective of this modeling exercise was to assess the suitability of on-line linkage between a water allocation model for river basins (RIBASIM) and a raster based agro-hydrological models developed in PC-Raster. Within the network model, the raster based model is represented as one node.

1.1 Model design

Within the current test model, each cell is autonomous, i.e. no interaction occurs between the cells. The model is subdivided into two phases, (1) a demand phase and (2) the allocation phase. The model preserves a simple water-balance of the soil. Due to the lengthy period of RIBASIM time steps, an time averaged value is used for parameters like groundwater drainage (see appendix 1).

1.1.1 The demand phase

At the start of the model run, the available precipitation is added to the soil reservoir. Using the reference evaporation and the cell-based crop factor, the actual evaporation is computed for every cell. This rate is overruled in case the water level equals the surface level, as in that situation the actual evaporation is set equal to the reference evaporation. The crop water requirement is set equal to the actual evaporation. Using the water available from the soil reservoir, the model tries to match the crop water requirement.

1.1.2 The allocation phase

In case the crop water requirement cannot be matched, the deficit will be transferred into a water demand from the network. During the allocation phase, an area averaged rate ill be supplied (if available). This averaged rate might return in local deviations from the desired situation as some cells might receive too much water while others might need more.
After the network allocated the water, this amount is added to the soil reservoir to compute percolation to the groundwater. Remaining water will be drained via the (sub)surface. The total drainage flow to the network comprises groundwater percolation as well as the (sub)surface flow.

1.1.3 Data sets

The model uses the following input time series:

- Precipitation [m]
- Reference-evaporation [m]
- Crop factor*

The following raster data sets (maps) are used:

- Area mask
- Vegetation types

And in a later stage:

- Soil water storage capacity*
- Soil surface*
- Soil water initial storage*
- Minimum soil water content to which the vegetation can extract water*
- Saturated conductivity (for deep percolation)*

The following data (time series) is exchanged with the network model:

- Drainage to the network
- Available water in the network
- Amount of water we actually use from the network

Parameters indicated with an '*' are variable per cell. The time series file with crop factors will contain as many columns as vegetation types (crops).

1.1.4 Time stepping

Agro-hydrological models are different from rainfall-runoff models. Due to the lengthy period of a time step in agro-hydrological models a fundamental difference exists in the processes being modeled within a time step. An agro-hydrological model computes a water demand (if any) per time step, which is passed to the network. The water offered by the network (to the agro-hydrological model) is used within the same time step.
A rainfall-runoff model however progresses in time after it passed the information to the network. This difference essentially means that an agro-hydrological model is composes of two sub-models, one for the demand phase and one for the allocation phase.

PC-raster, the raster based dynamic modeling language used for this exercise, has no notion of the length of a time step. Water balance models such as RIBASIM often work on a monthly or half monthly basis, having a different number of days every time step. Such variation in time step length can be handled, but the workaround is not easy and elegant.

1.2 Simple model test: Linkage to a network

Matlab scripts have been used to test the model and evaluate the results. The Matlab scripts emulate the control task of the RIBASIM network model, and call the PC-raster model where needed.

PC-Raster is not yet available as a callable library that can communicate in an on-line mode. Therefore, the executable is started for every time step, requiring every time step additional IO to read the actual status of the soil water reservoir and write the new status to file.

The control flow for the on-line connection is as follows:

```
For I=1 to last step do
      Network determines available water volume
      Pre-processor writes network data to file, using PC-raster format
      PC-raster demand model runs 1 time step (and writes data to file)
      Post processor reads PC-raster output and transfers demand to network node
      Network processes the demand
      Pre-processor writes new network data to file, using PC-raster format
      PC-raster allocation model runs 1 time step
      Post processor reads PC-raster output and transfers drainage to network node
      Network processes the drainage flow
End
```

This control loop requires every time step two complete starts and stops of the PC-raster executable; a time consuming processes.

Model	**Run-time (sec)**
On line (36 steps, 360 days)	14.06
Off line (36 steps, 360 days)	00.60

Note that this test model processes demand and allocation in one model and does not yet read and write the soil water balance to file. This means that the run-time will take twice as long, which is unacceptable in practice, unless the PC-raster kernel comes available as a callable library (a DLL with the PC-raster specific functions) which can be initiated once, can keep its (soil water balance) state and computes demands/drainage flows upon request. Another, but less flexible, option is direct implementation of the raster-based agro-hydrological model in F90 or C.

1.3 Test results

Two test runs have been conducted. The first run (Figure 1-1) concerns a uniform vegetation cover with crop factor 0.7. The second run (figure xx) uses three vegetation types with crop factor 0.78, 0.7, 0.64 respectively.

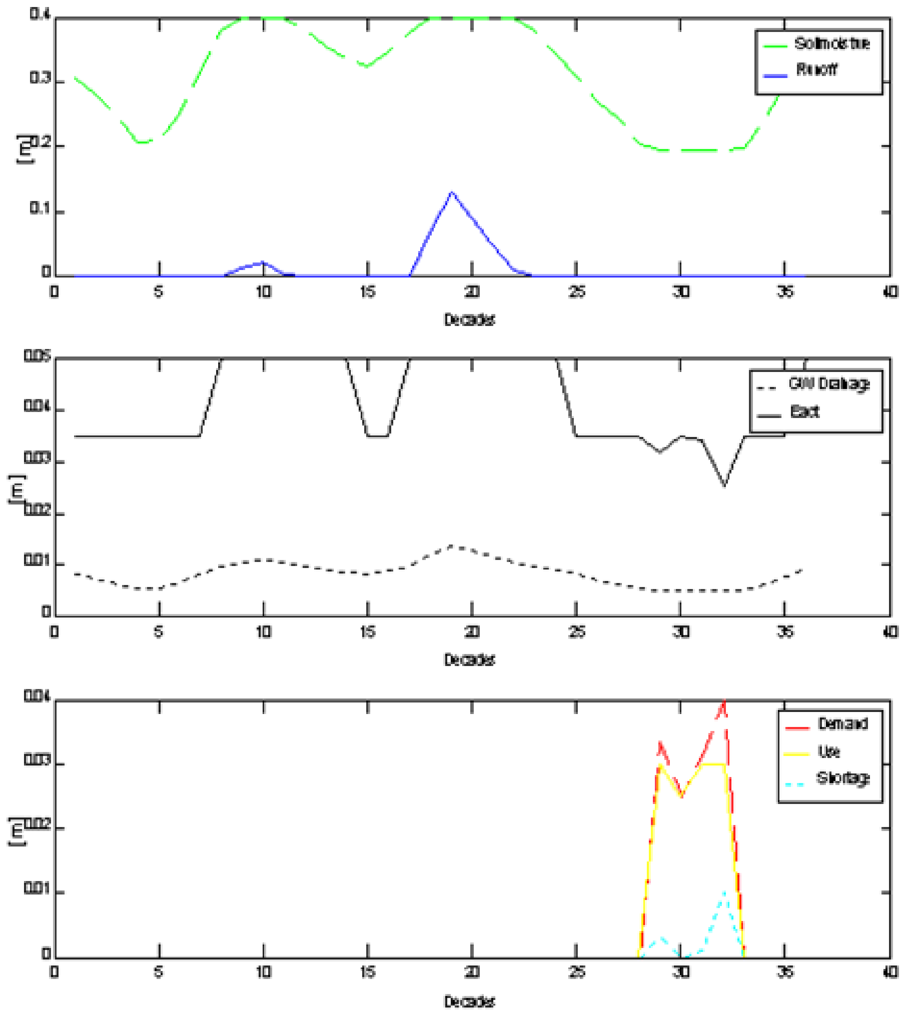


Figure 1-1 Results with uniform vegetation cover (cropfactor-0.7)

Total runoff:	0.381246
Total drainage:	0.291512
Total demand:	0.128908
Total use:	0.114938
Total shortage:	0.013971
Total E_{act}:	1.486029
Mean soilwater:	0.309315

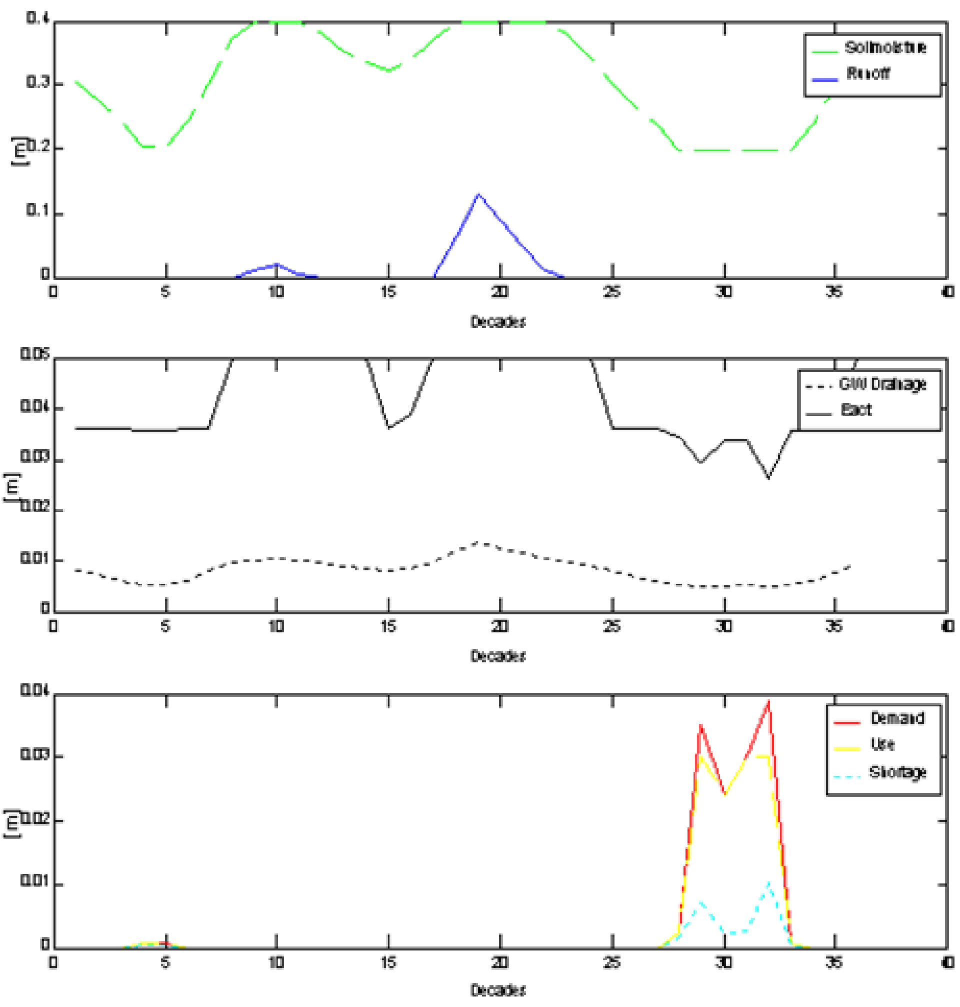


Figure 1-2 Results using three vegetation covers (crop factor 0.78, 0.7, 0.64 respectively)

Total runoff:	0.370935
Total drainage:	0.290090
Total demand:	0.132021
Total use:	0.118283
Total shortage:	0.025064
Total E_{act}:	1.502436
Mean soilwater:	0.308041

1.4 Suggestions

The following activities are foreseen to continue this work.
PC-raster

- turn the PC-raster kernel into a DLL with the specific PC-raster functions and data types;

Agro-hydrological models

- Develop a simple agricultural node (for RIBASIM) using the MUST model
- Develop an advanced agricultural node (for RIBASIM) using a layered soil model and Penman-Monteith evaporation (FAO-standard).
- Develop an advanced hydrological node, using a Digital Elevation Model, including water levels

1.5 Summary

- PC-raster enables easy and fast prototyping of distributed hydrological models;
- The limited IO functionality of PC-raster is a barrier to develop on-line linkages with a network model
- Currently on-line linkage is feasible, but requires substantial pre- and post processing and IO and is therefore very slow
- Variable time steps, used in RIBASIM, are currently difficult to implement in PC-raster
- Implementation of the PC-raster kernel as a library (a set of PC-raster specific functions and data-types) for application with common programming languages (VB, C++, Java, Fortran etc.) would facilitate the turnover from prototype into an operational model.

2 CropPlan: a tool for crop scheduling

The agricultural sector, and especially the irrigated agriculture, is one of the biggest sectors with respect to water withdrawal in a river basin. Therefore, proper planning of irrigation schemes and cropping patterns, and thus water demands, can be a useful method to improve the balance between water demand and water availability. A tool that assists in proper administration of the areas being planted, the time window of soil preparation, seeding, rotation schedules for irrigation etc. can assist in improving the overview of water demands by the agricultural sector.

However, such tool partly depends on the underlying agro-hydrological model. After an investigating of candidate agro-hydrological models by Schellekens [2001] it has been decided that the concept of the OMIS package (Operation and Management of Irrigation System, a DOS-package) is the most suitable candidate to become the basis for an the crop planning part of an agro-hydrological model.

As the design of this tool was written in Dutch [Sprengers 2001, Sprengers 2002] during the period that Delft Cluster reports were not forced to be English, this chapter will provide an extensive overview of this design.

2.1 Context and Requirements

The component will be developed to work with OMIS, AGWAT and ADIMO within the RIBASIM software package. The component will interact with the existing Ribasim database and will replace and extend the current data edit functionality for the agricultural model.
The tool needs:

- to provide a indication of the agricultural area at stake:
- to provide a cropping calendar having along the horizontal axis the time steps and along the vertical axis the area at stake
- to provide a water balance (demand and availability), that is automatically updated after modification of the cropping pattern
- to be launch able via Netter, the Network editor of RIBASIM/SOBEK
- to run stand-alone as well as in the RIBASIM/SOBEK environment
- to match at least the OMIS agricultural model, but preferably also others such as AGWAT and ADIMO.
- data needs to be easy accessible (e.g. via a tree view) and modifiable

2.2 The data model

2.2.1 Relevant data

The data is read from the RIBASIM database. The following data is relevant:
Voor 'Fixed irrigation' nodes:

- `Node ID`
- `Irrigated area [ha]: constant time series (per time step)`
- `Net demand [mm/day]: constant time series (per time step)`
- `Irrigation efficiency [%]`
- `Return flow percentage [%]`
- `Water management priority specification:`
  - `definition of part 1 of demand [%] (part 2 is the remaining demand)`
  - `priority for part 1 of the demand`
  - `priority for part 2 of the demand`

Voor 'Variable irrigation' nodes:

- Node ID
- Irrigated area for year type 1 [ha]
- Irrigated area for year type 2 [ha]
- Gross demand [mm/day] for year type 1: constant time series (p.timestep)
- Gross demand [mm/day] for year type 2: constant time series (p.timestep)
- Irrigation efficiency [%] for year type 1
- Irrigation efficiency [%] for year type 2
- Time series sequence index in expected rainfall time series file Exprain.Tms (in Fixed directory)
- Start of growing season (time step index)
- Return flow percentage of allocated water to surface water [%]
- Return flow percentage of unused rainfall to surface water [%]
- Water management priority specification:
  - definition of part 1 of demand [%] (part 2 is the remaining demand)
  - priority for part 1 of the demand
  - priority for part 2 of the demand

Voor 'Advanced irrigation' nodes:

- Node ID

Hydrology data

- Actual rainfall time series index in file Actrain.Tms
- Expected rainfall time series index in fixed data file Exprain.Tms
- Effective rainfall percentage [% of expected rainfall]
- Reference evapotranspiration time series index in fixed data file Refevapo.Tms
- Seepage for flood basin (paddy) kind of crops [mm/day]
- Soil moisture capacity [mm/dm]

Cultivation

- Technical irrigated area [ha]
- Technical irrigation efficiency [%]
- Technical irrigated area water re-use percentage [%]
- Technical irrigated area potential yield class index (defined in the fixed data file Pyldtcir.Dat)
- Technical irrigated area production costs class index (defined in the fixed data file Costtcir.Dat)
- Semi-technical irrigated area [ha]
- Semi-technical irrigation efficiency [%]
- Semi-technical irrigated area agricultural water re-use percentage [%]
- Semi-technical irrigated area potential yield class index (defined in the fixed data file Pyldstir.Dat)
- Semi-technical irrigated area production costs class index (defined in the fixed data file Coststir.Dat)

Operation

- Staggering period index
- Feedback on actual field water level: on/off switch
- Water shortage threshold for activation of agriculture water shortage reaction: increase of irrigation efficiency (mm/time step/irr.agric.area)
- Irrigation efficiency increase in case of agriculture water shortage [%]
- Water management

- Water management priority specification:
    - definition of part 1 of demand [%] (part 2 is the remaining demand)
    - priority for part 1 of the demand
    - priority for part 2 of the demand

2.2.2 The internal data structure

The data will be handled in VisualBasic 6 using the following data structures:

```
'Crop data
Public Type CropData
    nCrop As Integer             'nr. of crops
    CropFact() As Single         'cropfactor for each crop / timestep
    ExtLayer() As Single         'extra waterdem.for waterlayer/crop/timstp
    Rootdepth() As Single        'Rootddepth for each crop
    Treshold() As Single         'Treshold for each crop
    AllocPrior() As Integer      'Allocation priority for each crop
    PlantPeriod() As Integer     'length of plantperiod for each crop
    GrowPeriod() As Integer      'length of growingperiod for each crop
    CropType() As Integer        'flag wet/dry crop
    CropID() As Integer          'crop ID
    CropName() As String         'crop name
    IrrEff() As Single           'Overall irrigation efficiency (%)
    AreaEff() As Single          'Irrigation efficiency area (%)
End Type

'Crop planning data
Public Type CropPlan
    iAdvIrrNode As Integer       'ID of RIBASIM node
    nCult As Integer             'gross nr. of cultivations
    nCult0 As Integer            'net nr. of cultivations
    TotArea As Single            'total area cropping pattern
    Intensity As Single          'total cropintensity
    CultID() As Integer          'ID cultivation
    CultName() As String         'name cultivation
    TotCropReq() As Single       'total water demand per growing period
    CultCropID() As Integer      'index crop type per cultivation
    CultStart() As Integer       'start planting period per crop
    CultSeqID() As Integer       'index sequence planting period /crop
    CultSeq() As Integer         'seq. of cultivation,sorted at strt of grow
period
    CultArea() As Single         'area for each cultivation
    CultPercol() As Single       'percolation for each cultivation
    CultPresat() As Single       'presaturation for each cultivation
End Type

'Time data
Public Type TimeData
    nTim As Integer              'Nr. of timesteps
    nDays as Integer             'Nr/ of days in a time step
    AreaTime() As Integer        'Total cultivated area / time step
    XLabel() As String           'Name of X-label for graph / time step
End Type

'Data for all crops
Public MyCrops() As CropData     'Dimension: nCrop (/nTim)
Public MyPlan() As CropPlan      'Dimension: nCult
Public MyTime() As TimeData      'Dimension: nTim

'Waterbalance
Public Type DataSeries
    ErrRain() as Single          'effective rainfall
    DepInfl() as Single          'expected supply from river network (m3/s)
                                          '2-dim: 1=m3/s, 2=Mm3, 3=l/s/ha
    RefEvap() as Single          'reference crop evaporation
    ReqTim() as Single           'water demand per cultivation
    Demand() as Single          'water demand total area (m3/s)
                                         '2-dim: 1=m3/s, 2=Mm3, 3=l/s/ha
    Volume() as Single          'water volume reservoir (Mm3)
    QDepend() as Single         'expected discharge reservoir (Mm3)
    CropReq() as Single         'crop water demand (l/s/ha)
End Type

Public MyBalance() As DataSeries          'Dimension: nTim (/nCult)
```

Note: as ideas changed on the data model to be used, the implementation might take more advantage of collections.

2.2.3 The database

Figure 2-1 and Table 2-1 provide an overview of the database file structure associated to CropPlan. The database structure is an extension of the current RIBASIM database. The main extension is a new data file 'TOTPLAN' that contains the entire data set. The structure of TOTPLAN is similar to the structure of 'TOTPLAN.DRN', the data file used by the OMIS package, but is extended with a node reference to the RIBASIM network model.

All 'dat' and 'tms' files are ASCII-based input file and need to be read sequentially. All 'bin' files are binary random access file (Input and Output), while the 'drn' file is a sequential ASCII file.

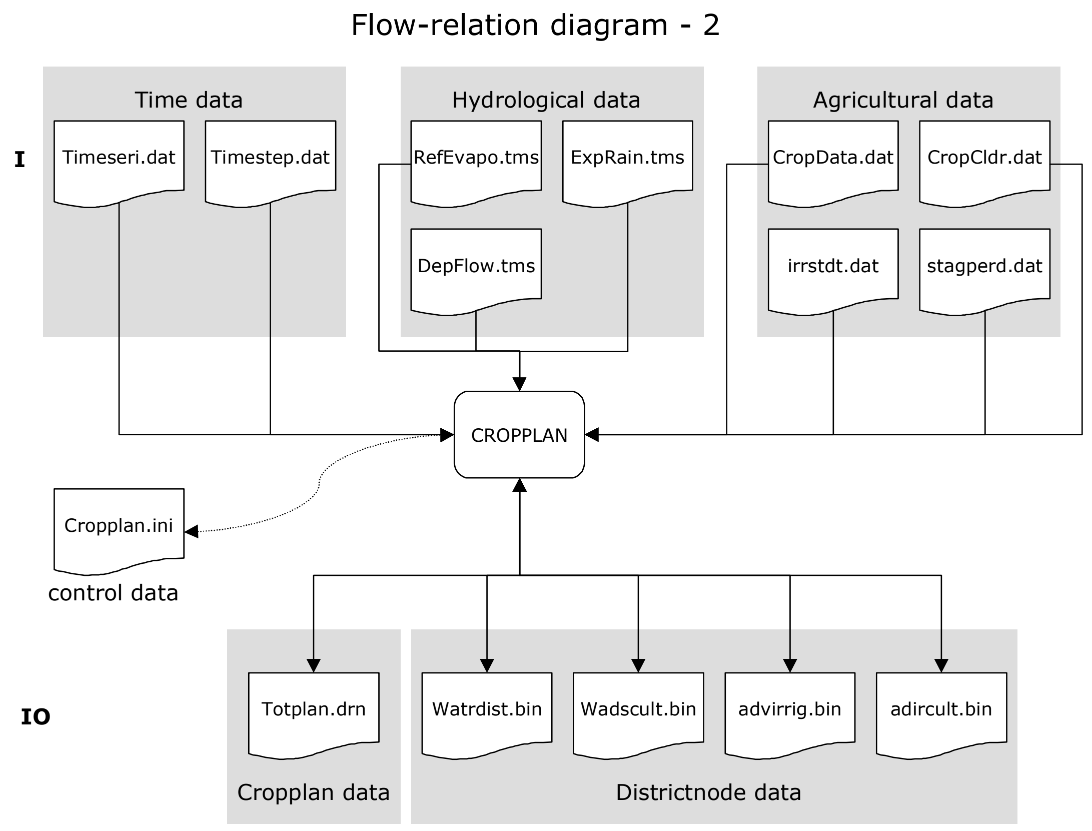


Figure 2-1 Overview of data file structure

Table 2-1 Overview of associated data files

Nr.	Description	File	I(O)	Read mode	Write mode
1	Time step data	..\fixed\timestep.dat	I	Sequential	-
2	Expected rainfall	..\fixed\exprain.tms	I	Sequential	-
3	Reference evaporation	..\fixed\refevapo.tms	I	Sequential	-

4	Time series file names, range	@timeseri.dat	I	Sequential	-
5	Crop data	..\fixed\cropdata.dat	I	Sequential	-
6	Cropping calendar data	..\fixed\cropcldr.dat	I	Sequential	-
7	Wadis water district data	@watrdist.bin	IO	Binary	Binary
8	Irr.season starting dates	..\fixed\irrsstdt.dat	I	Sequential	-
9	Staggering period	..\fixed\stagperd.dat	I	Sequential	-
10	Water dist.cultivation data	@wadscult.bin	IO	Binary	Binary
11	Advanced irrigation node data	@advirrig.bin	IO	Binary	Binary
12	Advanced irr.cutivation data	@adircult.bin	IO	Binary	Binary
13	Cropping pattern data	@totplan.drn	IO	Sequential	Sequential

2.2.4 IO and external interfaces

Note: the external interfaces might be improved to enable function calls from other programming environments (e.g. GIS macro languages).

Read/import ID's of the agricultural nodes in the network model

One or more ID's of the agricultural nodes can be passed during startup as arguments via the command line. In case of an empty argument list, all agricultural nodes will be displayed by default. The data of all relevant nodes will be read from the database and loaded into the CropPlan.

Read/import database files of the agricultural model

The data is read from the various data files using existing IO routines from DataEdit, RIBASIM, OMIS and AGWAT.

Write/export modified data to file

After data modification they need to replace the previous dataset in the files.

Report results to files:

A reporting functionality is foreseen to provide an overview of the results from the CropPlan.

- the following datasets can e viewed:
- demand (l/s/ha) for every cultivation at the inlet
- Total demand (Mm3 and l/s/ha) at the inlet
- Total demand (Mm3) per crop per season
- Water availability (Mm3) at inlet
- Waterbalance (Mm3) at the inlet.

Interaction with DataEdit

Note: this section might be due to modification if the invocation is enable via a direct function call to the COM-object.

While Netter invokes DataEdit, the default data editor of RIBASIM, DataEdit invokes the CropPlan. This requires that the script-file of DataEdit contains the following string reference to the CropPlan at the edit part of the agricultural nodes.

```
"..\\programs\cropplan.exe ..\programs \cropplan.ini ID1, ID2, ID3"
```

The initialization file cropplan.ini (see textbox) uses a Windows ini-settings structure and contains references to the RIBASIM databases as well as other settings of the editor.

```
[General]
RunMode=OMIS            'OMIS,AGWAT,ADIMO,?
GeneralIniFile=d:\projecten\q2987.22\tools\cropplan\General.ini
ReturnFile=d:\projecten\q2987.22\tools\cropplan\cropplan.rtn

[OMISNodedata]
Nodefile=
[OMISCropdata]
Cropdata=d:\projecten\q2987.22\data\omis\cropdata.drn
Cropfact=d:\projecten\q2987.22\data\omis\cropfact.drn
Waterlayer=d:\projecten\q2987.22\data\omis\extwtdmd.drn
Cropplan=d:\projecten\q2987.22\tools\cropplan\totplan.drn

[OMISDependable]
Refevap=d:\projecten\q2987.22\data\omis\refevapo.drn
Effrain=d:\projecten\q2987.22\data\omis\effrainf.drn
Estflow=d:\projecten\q2987.22\data\omis\pupflw.drn

[AGWATNodedata]
Nodefile=

[AGWATCropdata]
Cropdata=
Cropfact=
Waterlayer=
Cropplan=

[AGWATDependable]
Refevap=
Effrain=
Estflow=
```

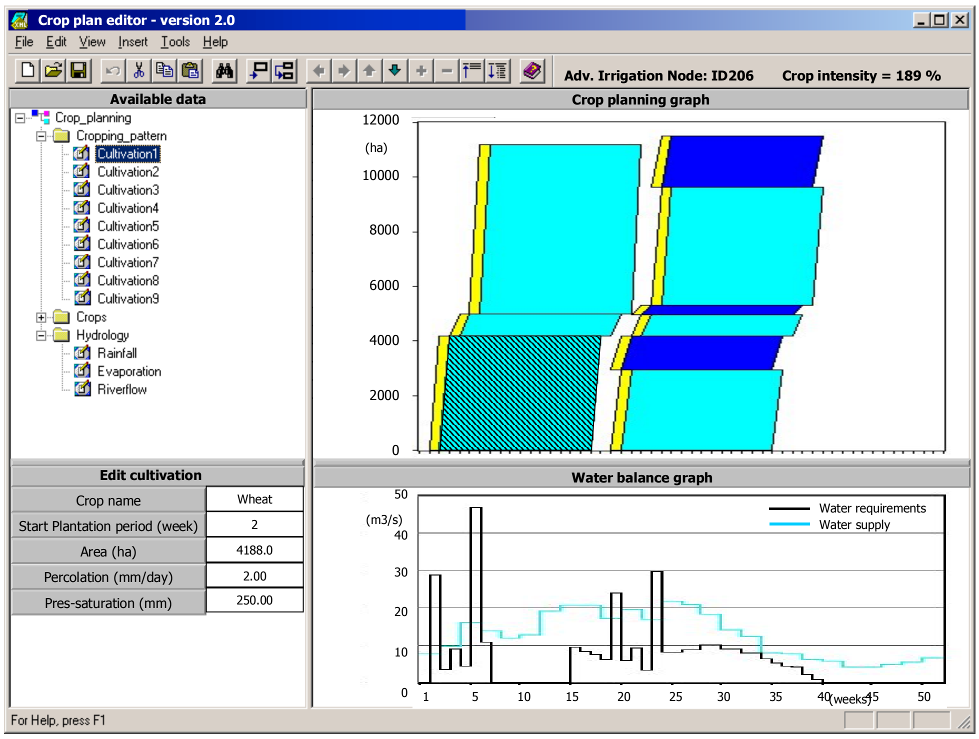
2.3 The User Interface

Four main functional groups can be identified, each of them taking place in a panel of the User Interface (see Figure 2-2)

Upper left panel: Display of agricultural nodes in a tree view
Every agricultural node has its own label in the tree view. When selecting this node, the associated cropping pattern will be shown.

Lower left panel: Display and edit of selected crop parameters
This panel enables viewing and modification of underlying model parameters and other data items.

Upper right panel: Display of crop pattern
The crop pattern is displayed using a parallelogram per cultivation. The slope of the parallelogram indicates the seeding/planting time, the horizontal length indicates the growing period. The height of the parallelogram indicates the surface area of the cultivation. On the left side, the land preparation/plantation time can be shown as a sub-parallelogram. The horizontal length typically is determined by the length of land preparation/plantation period (1 to 2 time steps). The horizontal axis indicates time while the vertical axis indicates the surface area expressed in 1000 hectares.



Crop name	Wheat
Start Plantation period (week)	2
Area (ha)	4188.0
Percolation (mm/day)	2.00
Pres-saturation (mm)	250.00

Figure 2-2 Example layout of CropPlan User Interface

Lower right panel: total water balance
The total water balance overview shows the total crop water demand and the total water availability over time of the entire cropped area. Time progresses along the horizontal axis, while the vertical axis contains the volumes in m^3/s or Mm^3.

Connections with other programs
The CropPlan is activated via the Netter program. Using the nodeID provided in the call any relevant crop data is retrieved from the RIBASIM database

Reporting of cropping patterns
After preparation of cropping patterns, overviews can be generated for reporting purposes. These overviews can be viewed via the 'view' menu or printed via the print option.

2.4 Computations

2.4.1 Water demand computation of wet land cultivations

The 'Van der Goor' method is used to represent the increase in cultivated area during the planting period. this method relates the non-linear increase of surface area to the constant flow of water. The progress in cultivation is described with:

$$Y = \{ e^{MT/S} / (e^{MT/S} - 1) \} * (1 - e^{-tM/S})$$

With :

Y = Fraction of the area cultivated at time stamp t (i.e. surface area that received the pre-saturation and water demand layer as well as compensation for seepage and evaporation)
t = Time [days]

S = Water demand for pre-saturation and water layer of paddy fields [mm]. Water is needed to maintain the water layer after the pre-saturation period is completed. This demand equals the evaporation + percolation [mm/day].
T = Duration of pre-saturation period [days] for the entire service area. Within this period, the pre-saturation should b completed.

The water demand is computed for a fixed period, e.g. one week. For each time step, the sequence of land preparation should be known to compute the associated water demand. It is assumed that, after seeding/planting, normal crop factors can be used to compute the cultivated part of the service area.

During land preparation the water demand is determined by:
- the replenishment of water for pre-saturation as well as the water layer for the area which is under preparation during time step dt
- evaporation and percolation of the area that has been prepared.

For the first period this comes down to;

$$\text{Total demand} = S{*}Y' + M * Y_{ep}'$$

S = Water demand for pre-saturation ad water layer of paddy fields [mm]
M = Water delivery to sustain the water layer after the pre-saturation is completed. This equals the sum of evaporation and percolation [mm/day].
Y' = Surface are that is cultivated at the end of the time step dt
Y_{ep}' = Factor indicating the cumulative evaporation and percolation rate.

The evaporation and percolation until time step t are given by:

$$\begin{aligned} M * A &= M * Y_{ep} \\ &= M * dYdt \\ Y_{ep} &= e^{MT/S} / (e^{MT/S} - 1) * \{ t + S/M * (e^{-tM/S} -1 \} \end{aligned}$$

The evaporation and percolation of a special section of the service area (saturated between t_0 and t) can be expressed as:

$$Y_{ep} = e^{MT/S} / (e^{MT/S} - 1) * \{ e^{-t0M/S} * (t-t_0-S/M) + S/M * e^{tM/S} \}$$

With:
Y = percentage of pre-saturated areas
t = Time (days)
t_0 = starting time of period (between t_0 and t) when pre-saturation should be completed.
T = duration of pre-saturation period [days]. Time frame within the pre-saturation should be completed.
M = water supply to sustain the water layer after pre-saturation is completed
S = water demand for pre-saturation of the paddy field
A = surface area under the line representing the percentage of pre-saturated area versus time.

2.4.2 Water demand computations of dry land cultivations

The water demand computation of dry land crops differs from the wet land crops, such ass paddy, in the sense that:
- the period of land preparation does not exist; in contrary a period exist in which the planting progresses;
- dry land crops only have limited rate of pre-saturation
- dry land crops do not have a water demand due to a water layer atop the soil

- dry land crops require an appropriate soil moisture condition in the root zone. A soil moisture condition at field capacity is desired. If the moisture condition decrease to wilting capacity, i.e. plants fading away, the actual evaporation rate need to be reduced resulting in drought damage. To prevent this damage, water delivery from rainfall or irrigation should keep the soil moisture content between a field capacity and a threshold (above wilting capacity).
- If soil moisture content is at field capacity and the soil reservoir capacity completely filled, excess water delivery will be drained off

Real time update of the water balance
Modification of the crop calendar immediately affects the water demand at one or more time steps.

Therefore, the water balance, shown in the lower right panel, is updated at real time based on abovementioned equations.

2.5 The main control flow

CropPlan accommodates a range of tasks. The associated actions are listed below.

0 Load frmMain
1 Check if Cropplan.exe is already running
2 Initialise

- 2.1 read commandline
- 2.2 read data from inifile
- 2.3 write crash - return code (optional)
- 2.4 load language
- 2.5 get helpfile
- 2.6 get and read datafiles
- 2.7 prepare data for cultivation file TOTPLAN.drn
- 2.8 load and set nodes of treeview
- 2.9 load first cropping pattern and data of first cultivation

3 Compose main window

- 3.1 SetLanguage
- 3.2 Display main window
- 3.3 Display treeview
- 3.4 Display edit window
- 3.5 Compute water demand
- 3.6 Display cropping pattern diagram
- 3.7 Display waterbalance graph

4 Wait for user action at main window

- 4.1 File-menu
  - 4.1.1 Save cropping pattern: save data to RIBASIM datafiles and TOTPLAN.drn
  - 4.1.2 Print cropping pattern data: sent data to printer
  - 4.1.3 Exit crop editor: display dialog when data has been changed and not saved
- 4.2 Edit-menu
  - 4.2.1 Cut: remove selected data from dataset
  - 4.2.2 Copy: copy selected data to clipboard from dataset
  - 4.2.3 Paste: paste selected data from clipboard to dataset
  - 4.2.4 Add cultivation: add new cultivation to dataset
  - 4.2.5 Edit cropping pattern: open file TOTPLAN.drn in edit window using spreadsheet control
- 4.3 View-menu
  - 4.3.1 View water requirements per cultivation: get data, open view window
  - 4.3.2 View water requirements per crop season: get data, open view window
  - 4.3.3 Total irrigation requirement: get data, open view window

    - 4.3.4 View water availabillity: get data, open view window
    - 4.3.5 Total waterbalance: get data, open view window
  - 4.4 Tools-menu
    - 4.4.1 Options: open dialog to select the unit of Y-axis in the time-waterbalance graph: m3/s or Mm3 per timestep
  - 4.5 Help-menu
    - 4.5.1 Help on crop editor: opens cropplan.chm helpfile which has HTML functionality and contains a content and keyword list
    - 4.5.2 About: displays dialog window with userinfo, license info, current version
  - 4.6 Treeview
    - 4.6.1 Click tree node to display/hide list of cultivations
    - 4.6.2 Generate event to update fields in edit-window
    - 4.6.3 Rename existing acultivation
    - 4.6.4 Add new cultivation
    - 4.6.5 Delete existing cultivation
  - 4.7 Edit-window
    - 4.7.1 Enter data in edit fields
    - 4.7.2 Generate event to compute water demand and update waterbalance graph
    - 4.7.3 Generate event to update crop calendar graph

5 Compute water demand
  - 5.1 Update water demand for each cultivation per timestep
  - 5.2 Compute total water demand per timestep
  - 5.3 Compute total water demand per crop season
  - 5.4 Compute total irrigation requirement

6 User selects File|Exit: unLoad frmMain
  - 6.1 Display dialog with user prompt “Data has been changed. Do you want to save the changes ?” The user may select one of the buttons ‘Yes’ or ‘No’
    - 6.1.1 Yes-button: save file TOTPLAN.drn and AGWAT *.bin files (see 4.1.1)
    - 6.1.2 No-button: no action
  - 6.2 Close all open data files
  - 6.3 Write return code to return code file
  - 6.4 End application using End-statement

2.6 Other items

Aspects covered in the Dutch design report, but left out of this report are amongst others the detailed description to implement the various functions, the associated data flow diagrams, the implementation plan (incl. budgeting) and the test plan.

3 References

Schellekens, J. 2001
Watervraag/Districts modellen, WL Delft Hydraulics memo 10 oktober 2001, Q2907.12

Sprengers, C.J. 2001
Gewasplanningsmodule OMIS-CP 2.0 Ontwerp, WL Delft Hydraulics concept rapport Q2987.22, December 2001

Sprengers, C.J. 2002
RIBASIM ontwerp gewaseditor Detail, functioneel en technisch ontwerp, WL Delft Hydraulics concept rapport Q2907.12, November 2002

Appendix: Agro-hydrological test model – PCraster and Matlab scripts

PC-raster scrips

```
Demand = -1 * Soilm * idxx;

# ALLOCATION PHASE
############################################################
# At this time we start asking (or not) water from the network..
# Assuming that we do not know what happens in detail we ask for
# an area average demand...
 report WeAskTS = maptotal(Demand)/maparea(Demand);
# Now see what the network has to offer...
 report WeUseTS = maptotal((if (WeAskTS < Supply then WeAskTS else
Supply)))/maparea(Soilm);

# Now update the soil water...
 Soilm += WeUseTS;

# Negative values indicate a Shortage.   our new Demand/Shortage
 idxx = scalar(Soilm < 0);
 $7 Demand = -1 * Soilm * idxx;

# If the Demand has not been met in the allocation phase we have to
# get rid of the negative soil water values...
 idxx = scalar(Soilm > 0);
 Soilm = Soilm * idxx;

# We are done. Now add SoilMin to Soilm again...
 Soilm += SoilMin;

# Now calculate groundwater drainage
# Use a linear approuch, just as a test...
# First estimation....
 GWdrain = Ksat * Soilm/Soilcap;
# Now calculate using average soil water content...
 GWdrain = Ksat * ((Soilm - GWdrain + Soilm) * 0.5)/Soilcap;
 Soilm = if(Soilm - GWdrain < 0 then 0 else Soilm - GWdrain);

# Skimm of excess water (> storage capacity) add to drainage/runoff
 $7 Runoff = if(Soilcap -Soilm < 0 then Soilm -Soilcap else 0);
 $7 Soilm = if (Soilm > Soilcap then Soilcap else Soilm);
 report SoilmTS = maptotal(Soilm)/maparea(Soilm);
 report RunoffTS = maptotal(Runoff)/maparea(Runoff);
 report GWdrainTS = maptotal(GWdrain)/maparea(GWdrain);

# Reduce Actal ET according to shortage
 $7 Eact = Eact - Demand;
 report EactTS = maptotal(Eact)/maparea(Eact);

# These are the time series for interacting with the network
 $7 Shortage = Demand;
 report ShortTS=maptotal(Shortage)/maparea(Shortage);

 Drainage=Runoff + GWdrain;
 report WeGiveTS=maptotal(Drainage)/maparea(Drainage);

# Set some stuff back to zero...
 Demand = 0;
```

Matlab scripts

```
function mat=tss2mat(fname)
% function mat=tss2mat(fname)
% tss2mat takes a filename containing pcraster time series
% data and stores it in mat.
%
% author:   J. Schellekens 2002
% Version:  1.0
% Date:     01/02/2002
% Bugs:     the description is not implemented yet!

if nargin <1
    error('syntax: tss2mat(fname)');
end

fp = fopen(fname,'r');

if fp == -1
    error(sprintf('could not open file: %s',mfname));
end

% skip the first comment line..
fgets(fp);
tstr = fgets(fp);
% get nr of columns...
cols = sscanf(tstr,'%d',1);

% skip columns description lines...
for i=1:cols
    fgets(fp);
end

% now read the actual data
mat = fscanf(fp,'%f',[cols inf])';


function tss=mat2tss(mat,filename,descript)
% function tss=mat2tss(mat,filename,descript)
%
% mat2tss take a matrix (assuming time in column 1) and writes
% it out in filename in the PCraster time series format.
% Optional an string array with descriptions
% for each column can be given.
%
% author:   J. Schellekens 2002
% Version:  1.0
% Date:     01/02/2002
% Bugs:     the description is not implemented yet!

if nargin >= 2
    ssize = size(mat);
else
    error('syntax: mat2tss(mat,filename,descript)');
end

fp = fopen(filename,'w');
if fp == -1
    error(sprintf('Could not open file: %s',filename));
end

% print comment line
fprintf(fp,'Matlab generated tss file\n');
% print number of columns
fprintf(fp,'%d\n',ssize(2));
for i=1:ssize(2)
    fprintf(fp,'data column %d\n',i);
end
for i=1:ssize(1)
    fprintf(fp,'%g\t',mat(i,:));
    fprintf(fp,'\n');
end

fclose(fp);
```

```
% Script to (try and) run the PCraster RIBASM node using
% one line data exchange for water demand

% data is exchanged using three files in the network directory
% avail.tss   -> what we (the network) have to offer
% use.tss     -> what the node uses
% dischar.tss -> what we get back from the node

% in order to use the same time-series as the off-line version those time series

% are read from file and store (per timestep) in the ri_inp dirs.


'% OF COURSE SOILM.map SHOULD BE READ AND WRITTEN !!!!! action needed !!'

nworkdir = 'ri_network';
restdir =  'ri_out';
inpdir = 'ri_inp';
timeline = [1:36]';
avail = 0.03; % set this fixed at present

crop = tss2mat('timein/cropf.tss');
eref = tss2mat('timein/eref.tss');
precip = tss2mat('timein/precip.tss');

% data is collected in these variables...
dis = zeros(36,2);
use = zeros(36,2);
runoff = zeros(36,2);
demand = zeros(36,2);
soilm = zeros(36,2);
short = zeros(36,2);
eact = zeros(36,2);
drain = zeros(36,2);

for i=1:max(size(timeline))
    fprintf(2,'timeline: %d avail: %f\n',timeline(i),avail);
    % here we make the avial file for pcraster
    mat2tss([1 avail],sprintf('%s/avail.tss',nworkdir));

    % prepare input tiem series for PCraster here...
    mat2tss([1 precip(i,2)],sprintf('%s/precip.tss',inpdir));
    mat2tss([1 eref(i,2)],sprintf('%s/eref.tss',inpdir));
    mat2tss([1 crop(i,2)],sprintf('%s/cropf.tss',inpdir));

    % now run the PCraster excecutable...
    pcrrun(sprintf('-f rib10d.mod %s %s %s %s %s 1
report',nworkdir,'mapin',restdir,inpdir,restdir))

    % Start reading the return values from the network
    dis(i,:) = tss2mat(sprintf('%s/dischar.tss',nworkdir));
    use(i,:) = tss2mat(sprintf('%s/use.tss',nworkdir));

    % also read the extra output in the ri_out dir
    runoff(i,:) = tss2mat(sprintf('%s/runoff.tss',restdir));
    runoff(i,1) = i;
    demand(i,:) = tss2mat(sprintf('%s/ask.tss',restdir));
    demand(i,1) = i;
    soilm(i,:) = tss2mat(sprintf('%s/soilm.tss',restdir));
    soilm(i,1) = i;
    short(i,:) = tss2mat(sprintf('%s/shortage.tss',restdir));
    short(i,1) = i;
    eact(i,:) = tss2mat(sprintf('%s/eact.tss',restdir));
    eact(i,1) = i;
    drain(i,:) = tss2mat(sprintf('%s/gwdrain.tss',restdir));
    drain(i,1) = i;

end

% now dump the complete time-series in the directories...
```