PCRasterModflow

PCRasterModflow user manual

PCRasterModflow: PCRasterModflow user manual

Published This document was generated 2011-03-22.

Table of Contents

Chapter 1. Introduction

PCRaster is a Geographical Information System using a raster based approach to analyse spatial data. The included modelling language allows to develop iterative spatial-temporal environmental models. Detailed informations about PCRaster are available at http://pcraster.geo.uu.nl.

MODFLOW simulates the three-dimensional groundwater flow through a porous medium by using a finite-difference method. It is developed by the U.S. Geological Survey and widely used for groudwater modelling. Amongst others impacts of rivers, drains and wells can be simulated. For detailed informations about the physical and mathematical concepts we refer to the documentation at the MODFLOW website (http://water.usgs.gov/nrp/gwsoftware/modflow2000/modflow2000.html) and the user manuals (e.g. TWRI 6-A1, Open-File Report 00-92, both online available).

The PCRasterMODFLOW extension enables to embed the groundwater flow calulations in dynamic models described with the PCRaster modelling languages.

Chapter 2. Quickstart

The PCRasterModflow subdirectory of the demo directory includes a PCRcalc and a Python script both rebuilding the same example problem given in the user guide of MODFLOW-2000 (Open file report 00-92).

Run the example with

```
pcrcalc -f example.mod
```

or

```
python2.5 example.py
```

Chapter 3. Usage in PCRCalc

This chapter describes the usage of the PCRasterMODFLOW extension. The first step is to create one single extension object in the script's inital section:

```
initial
...
# Construct MODFLOW extension object
object mf = PCRasterModflow::initialise();
```

This will initialise the data structures used in the extension. All operations described in the package sections will refer to the object **`mf`**.

Warning

Operations specifying MODFLOW input data currently return a boolean map containing false values, this behaviour may change in the future. It is strongly recommended to keep the coding style as shown below (i.e. res = mf::operation();) and advised not to use the result in oncomming operations.

The next step is the grid specification using the operations described in the DIS package. This must be done before any other package is defined. Afterwards packages can be defined in arbitrary order. For a MODFLOW simulation at least the DIS, BAS and BCF package must be specified.

The DIS, BAS, BCF and a solver package must be set in the initial section of a script. Stress packages (RIV, DRN, RCH and WEL) can be activated and modified in the dynamic section.

Note

One timestep in PCRaster represents one stress period in MODFLOW.

In the following non-spatial arguments are written capitalised.

3.1. Setting MODFLOW's input data

This section describes the provided operations for the input data of MODFLOW and how to retrieve the ouput of MODFLOW.

3.1.1. DIS package

The DIS package specifies the grid used for the groundwater model. The grid specification must start with the bottom layer. Afterwards confined/unconfined layer can be added. The maximum number of layers is 999.

The operation for the specification of the bottom layer is

```
res = mf::createBottomLayer(bottomElevation, topElevation);
```

where

bottomElevation — is the name of a spatial, scalar PCRaster map containing the bottom elevation values of the layer. The map must not contain missing values;

topElevation — is the name of a spatial, scalar PCRaster map containing the top of layer elevation values. The map must not contain missing values.

The operation to add a layer is

```
res = mf::addLayer(elevation);
```

where

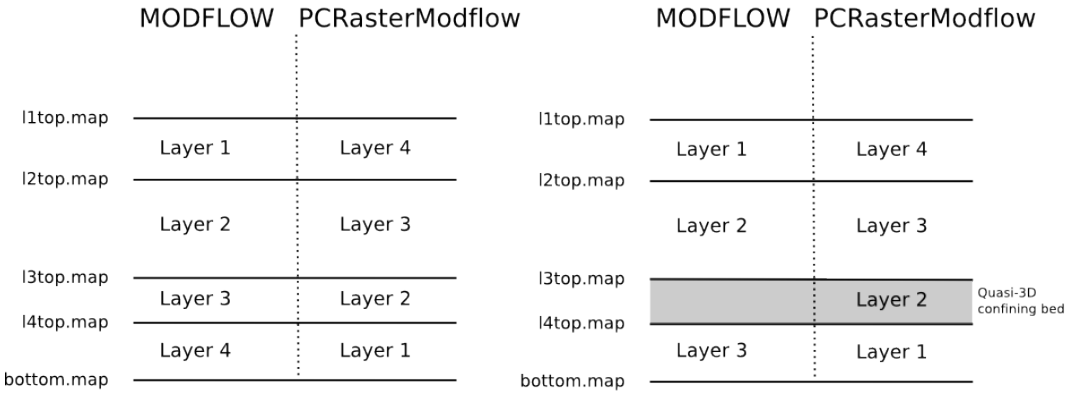
elevation is the name of a spatial, scalar PCRaster map containing the top of layer elevation values. The map must not contain missing values.

The operation to add a confined layer is

```
res = mf::addConfinedLayer(elevation);
```

where

elevation is the name of a spatial, scalar PCRaster map containing the top of layer elevation values. The map must not contain missing values.



Layer numbering in MODFLOW and the PCRasterMODFLOW extension

The figure shows two grid specifications. The left side of the figure represents a four layer system that is specified in a PCRcalc script as follows:

```
res = mf::createBottomLayer(bottom.map, l4top.map);
res = mf::addLayer(l3top.map);
res = mf::addLayer(l2top.map);
res = mf::addLayer(l1top.map);
```

The right side of the figure shows a three layer system with a confining bed below layer 2:

```
res = mf::createBottomLayer(bottom.map, l4top.map);
res = mf::addConfinedLayer(l3top.map);
res = mf::addLayer(l2top.map);
res = mf::addLayer(l1top.map);
```

The PCRasterMODFLOW extension uses an opposite layer numbering to the MODFLOW convention. Furthermore quasi-3D confining beds obtain a layer number. Layer numbering always starts with layer number 1 for the bottom layer and increases for each added confined or unconfined layer.

Except for setting the conductivity values all commands operate on layers which are not specified as confining beds. Attempts to set or retrieve values from confining beds will result in an error.

3.1.1.1. Optional operation

The options for the DIS package can be specified with

```
res = mf::setDISParameter(ITMUNI,LENUNI,PERLEN,NSTP,TSMULT,SSTR);
```

where

ITMUNI indicates the time unit (0: undefined, 1: seconds, 2: minutes, 3: hours, 4: days, 5: years);

LENUNI indicates the length unit (0: undefined, 1: feet, 2: meters, 3: centimeters);

PERLEN is the duration of a stress period;

NSTP is the number of iterations;

TSMULT is the multiplier for the length of the successive iterations;

SSTR 0 - transient, 1 - steady state. If the simulation is set to transient, primary and secondary storage coeffiecents must be set in the BCF package.

All input values are non spatial values. If this operation is not used the simulation will be set to the default values of (undefined, undefined, 1.0, 1, 1.0 , 1).

3.1.2. BAS package

The BAS package specifies the input for the basic package. It is necessary both to set the boundary and the starting head values for each layer. If a command is not given vor a layer the cells are set to the default values of 0 (inactive cell) and a initial head value of 0.

The operation for the boundary specification is

```
res = mf::setBoundary(boundaryValues, LAYER);
```

where

boundaryValues is the name of a spatial, nominal PCRaster map (-1: cell has constant head, 0: cell is inactive, 1: cell is active) and

LAYER is the MODFLOW layer number the map values will be assigned to.

The operation for the initial head value specification is

```
res = mf::setInitialHead(initialValues, LAYER);
```

where

initialValues is the name of a spatial, scalar PCRaster map containing the head values and

LAYER is the layer number the map values will be assigned to.

3.1.2.1. Optional operation

The options for the BAS package can be specified with

```
res = mf::setNoFlowHead(VALUE);
```

where

VALUE is the scalar, non-spatial value of the head to be assigned to all no flow cells (HNOFLO).

If this operation is not used the value will be set to a default value of -999.99.

3.1.3. BCF package

The BCF package specifies the input for the block-centered flow package.

The operation for the specification of the conductivity values is

```
res = mf::setConductivity(LAYTYPE, hCond, vCond, LAYER);
```

where

LAYTYPE contains the combined code for the method of computing interblock conductance (left digit; 0 or blank: harmonic mean (MODFLOW-88), 1: arithmetic mean, 2: logarithmic mean, 3: arithmetic mean of saturated thickness and logarithmic mean hydraulic conductivity) and the layer type (LAYCON, right digit; 0: confined, 1: unconfined, 2: confined/ unconfined (transmissivity constant), 3: confined/ unconfined (transmissivity varies));

hCond is the name of a spatial, scalar PCRaster map containing the horizontal conductivity values;

vCond is the name of a spatial, scalar PCRaster map containing the vertical conductivity values and

LAYER is the layer number the map values will be assigned to.

Annotation: For calculations in MODFLOW only vertical hydraulic conductivity values for quasi-3D confining beds are used. Horizontal conductivity values must also be specified for those layer due to technical reasons.

3.1.3.1. Transient simulations

If the simulation state is set to transient the specification of the storage coefficients is required. The operation is

```
res = mf::setStorage(primaryStorage, secondaryStorage, LAYER);
```

where

primaryStorage is the name of a spatial, scalar PCRaster map containing the primary storage values;

secondaryStorage is the name of a spatial, scalar PCRaster map containing the secondary storage values and

LAYER is the layer number the map values will be assigned to.

3.1.3.2. Wetting capability

To activate the wetting capability the following operations must be used. The non-spatial parameters are set with

```
res = mf::setWettingParameter(WETFCT, IWETIT, IHDWET);
```

where

WETFCT is the factor that is included in the calculation of the head that is initially established at a cell when that cell is converted from dry to wet;

IWETIT is the iteration interval for attempting to wet cells. Wetting is attempted every IWETIT iteration. This applies to outer iterations and not inner iterations. If IWETIT is 0, the value is changed to 1;

IHDWET flag that determines which equation is used to define the initial head at cells that become wet (0: h = BOT + WETFCT (h_n - BOT), 1: h = BOT + WETFCT (THRESH)).

The wetting threshold and flag values are specified with

```
res = mf::setWetting(map, LAYER);
```

where

map is the name of a spatial, scalar PCRaster map and

LAYER is the layer number the map values will be assigned to.

3.1.3.3. Optional operations

The head value that is assigned to cells that are converted to dry during a simulation (HDRY) can be specified with

```
res = mf::setDryHead(VALUE);
```

where

VALUE is the scalar, non-spatial head value.

If this operation is not used the value will be set to a default value of -999.9.

The variable containing the horizontal anisotropy factor (TPRY) can be specified with

```
res = mf::setHorizontalAnisotropy(VALUE, LAYER);
```

where

VALUE is the scalar, non-spatial horizontal anisotropy value.

LAYER is the layer number the map values will be assigned to.

If this operation is not used for a specific layer the value will be set to a default value of 1.0 (isotropic conditions).

3.1.4. RIV package

The RIV package specifies the contribution or drainage of the rivers to/from the aquifer. To enable the river package use

```
res = mf::setRiver(head, bottom, conductance, LAYER);
```

where

head [L]	is the name of a spatial, scalar PCRaster map containing the head values;
bottom [L]	is the name of a spatial, scalar PCRaster map containing bottom values;
conductance [L^2T^{-1}]	is the name of a spatial, scalar PCRaster map containing the conductance values. The cell values must contain either 0 (no river) or the conductance value of the corresponding river cell;
LAYER [-]	is the layer number the map values will be assigned to.

3.1.5. RCH package

The RCH package specifies the areally distributed recharge. To enable the recharge package either use

```
res = mf::setRecharge(recharge, NRCHOP);
```

where

recharge [LT^{-1}]	is the name of a spatial, scalar PCRaster map containing the recharge flux and
NRCHOP [-]	is the recharge option code (1 - recharge is only to the top grid layer, 3 - recharge is applied to the highest active cell in each vertical column),

or to indicate the layer number where recharge is applied to use

```
res = mf::setIndicatedRecharge(recharge, layer);
```

where

recharge [LT^{-1}]	is the name of a spatial, scalar PCRaster map containing the recharge flux and
layer [-]	is the name of a spatial, nominal PCRaster map containing layer number variables defining the layer in each vertical column where recharge is applied.

3.1.6. WEL package

The WEL package specifies the the use of wells in a simulation. To enable the well package use

```
res = mf::setWell(rates, LAYER);
```

where

rates	is the name of a spatial, scalar PCRaster map containing the well values. The cell values must contain either 0 (no well), positive (injection) or negative (pumping) volumetric rates and
LAYER	is the layer number the map values will be assigned to.

Only layer containing wells need to be assigned, the remaining layer will be set to 0.

3.1.7. DRN package

The DRN package specifies the use of drains in a simulation. To enable the drain package use

```
res = mf::setDrain(elevation, conductance, LAYER);
```

where

elevation [L] is the name of a spatial, scalar PCRaster map containing drain elevation values;

conductance [L^2T^{-1}] is the name of a spatial, scalar PCRaster map containing drain conductance values and

LAYER [-] is the layer number the map values will be assigned to.

Water seepage into a drain is calculated if the conductance value in a cell is larger than zero.

3.1.8. Solver packages

One of the following solvers must be specified for a model run. All arguments for the solver are non-spatial values.

3.1.8.1. PCG package

The preconditioned conjugate-gradient package can be enabled with

```
res = mf::setPCG(MXITER, ITERI, NPCOND, HCLOSE, RCLOSE,
RELAX, NBPOL, DAMP);
```

where

MXITER is the maximum number of outer iterations;

ITERI is the number of inner iterations;

NPCOND 1 - Modified Incomplete Cholesky, 2 - Polynomial matrix conditioning method;

HCLOSE is the head change criterion for convergence;

RCLOSE is the residual criterion for convergence;

RELAX is the relaxation parameter used with NPCOND = 1;

NBPOL indicates whether the estimate of the upper bound on the maximum eigenvalue is 2.0 and

DAMP is the damping factor.

3.1.8.2. SOR package

The slice-successive overrelaxation package can be enabled with

```
res = mf::setSOR(MXITER, ACCL, HCLOSE);
```

where

MXITER is the maximum number of iterations allowed in a time step;

ACCL is the acceleration variable and

HCLOSE the head change criterion for convergence.

3.1.8.3. SIP package

The strongly implicit procedure package can be enabled with

```
res = mf::setSIP(MXITER, NPARAM, ACCL, HCLOSE, IPCALC, WSEED);
```

where

MXITER is the maximum number of times through the iteration loop in one time step;

NPARAM is the number of iteration variables to be used;

ACCL is the acceleration variable;

IPCALC 0 - the seed entered by the user will be used, 1 - the seed will be calculated at the start of the simulation, and

WSEED is the seed for calculating iteration variables.

3.1.8.4. DSP package

The direct solver package can be enabled with

```
res = mf::setDSP(ITMX, MXUP, MXLOW, MXBW, IFREQ, ACCL, HCLOSE);
```

where

ITMX is the maximum number of iterations each time step;

MXUP is the maximum number of equations in the upper part of the equations to be solved;

MXLOW is the maximum number of equations in the lower part of equations to be solved;

MXBW is the maximum band width plus 1 of the [AL] matrix;

IFREQ is flag indicating the frequency at which coefficients in [A] change;

ACCL is a multiplier for the computed head change for each iteration and

HCLOSE is the head change closure criterion.

3.2. Retrieving MODFLOW's output

Afterwards the specification of the grid MODFLOW can be called with

```
res = mf::run();
```

After a successful MODFLOW run the results of a stress period can be reported with the commands described in this section. If the run fails the end of the MODFLOW list file is displayed, the execution of the script ends.

The head and boundary values are retrieved automatically and must not be set again for the next stress period. Applying the operations to layer specified as quasi-3D confining beds will result in an error.

```
head.map = mf::getHeads(LAYER);
```

head.map is a scalar PCRaster map containing the resulting head values of the layer *layer*. Cells converted to dry obtain the data type missing value.

```
river.map = mf::getRiverLeakage(LAYER);
```

river.map is a scalar PCRaster map containing the resulting river cell-by-cell flow values (in $[L^3T^{-1}]$) of the layer *layer*.

```
rch.map = mf::getRecharge(LAYER);
```

rch.map is a scalar PCRaster map containing the resulting recharge cell-by-cell flow values (in $[L^3T^{-1}]$) of the layer *layer*.

```
drn.map = mf::getDrain(LAYER);
```

drn.map is a scalar PCRaster map containing the resulting drain cell-by-cell flow values (in $[L^3T^{-1}]$) of the layer *layer*.

Chapter 4. Usage in Python

This chapter describes the usage of the PCRasterMODFLOW extension within Python scrips. Using the extension requires the import of the extension module, specification of the geographic extents and the creation of an extension object:

```
from pcraster import *
from PCRasterModflow import *

setclone("clone.map")
mf = initialise(clone())
```

This will initialise the data structures used in the extension. All operations will operate on the object **`mf`**.

The differences of Python operations to the PCRcalc syntax is minimal and therefor only brief explained: The Python extension operations have the same name and take the same arguments as the PCRcalc operations. Setting for example boundary values in PCRCalc is done by

```
res = mf::setBoundary(boundaryValues, LAYER);
```

and in Python by

```
mf.setBoundary(boundaryValues, LAYER)
```

where boundaryValues is either a string holding a filename to a spatial, scalar PCRaster map or a variable holding an object returned by the readmap operation.

See the Python example script for further operations.

Chapter 5. Examples

Both scripts describe the example problem given in the MODFLOW-2000 user guide.

5.1. PCRcalc model script

```
# This example script demonstrates the use of the
# PCRasterModflow extension. It recreates the example
# problem given in the user guide of MODFLOW-2000
# (Open file report 00-92)
#
# To run the script use
# pcrcalc -f example.mod

binding
  elev = elev.map;
  bottom = bottom.map;
  l1_top = l1_top.map;
  l2_top = l2_top.map;
  l3_top = l3_top.map;
  l4_top = l4_top.map;
  l5_bound = l5_bound.map;
  l3_bound = l3_bound.map;
  l1_bound = l1_bound.map;
  head = head.map;
  hcond_l5 = hcond_l5.map;
  hcond_l3 = hcond_l3.map;
  hcond_l1 = hcond_l1.map;
  vcond_l1=vcond_l1.map;
  vcond_l2=vcond_l2.map;
  vcond_l3=vcond_l3.map;
  vcond_l4=vcond_l4.map;
  vcond_l5=vcond_l5.map;
  rch = rch.map;
  drain_cond = drain_cond.map;
  drain_elev = drain_elev.map;
  wel_l1 = well_l1.map;
  wel_l3 = well_l3.map;
  wel_l5 = well_l5.map;

areamap
  clone.map;

timer
  1 1 1;

initial
  object mf = PCRasterModflow::initialise();

  # do not use result maps as input in oncomming operations!

  # grid specification
  res = mf::createBottomLayer(bottom, l1_top);
  res = mf::addConfinedLayer(l2_top);
  res = mf::addLayer(l3_top);
  res = mf::addConfinedLayer(l4_top);
```

```
  res = mf::addLayer(elev);

  # simulation parameter
  res = mf::setDISParameter(1, 0, 86400, 1, 1, 1);

  # adding the boundary conditions to the MODFLOW model
  res = mf::setBoundary(l1_bound, 1);
  res = mf::setBoundary(l3_bound, 3);
  res = mf::setBoundary(l5_bound, 5);

  # adding the initial values to the MODFLOW model
  res = mf::setInitialHead(head, 1);
  res = mf::setInitialHead(head, 3);
  res = mf::setInitialHead(head, 5);

  # adding the conductivities to the MODFLOW model
  res = mf::setConductivity(0, hcond_l1, vcond_l1, 1);
  res = mf::setConductivity(0, hcond_l1, vcond_l2, 2);
  res = mf::setConductivity(0, hcond_l3, vcond_l3, 3);
  res = mf::setConductivity(0, hcond_l3, vcond_l4, 4);
  res = mf::setConductivity(1, hcond_l5, vcond_l5, 5);

  # solver package
  res = mf::setSIP(50, 5, 1.0, 0.001, 0, 0.001);

  res = mf::setDryHead(1E30);

  # adding the drain
  res = mf::setDrain(drain_elev, drain_cond, 5);

  # specifying the recharge
  res = mf::setRecharge(rch, 1);

  # adding well
  res = mf::setWell(wel_l1, 1);
  res = mf::setWell(wel_l3, 3);
  res = mf::setWell(wel_l5, 5);


  # execute MODFLOW modelling engine
  res = mf::run();

  # retrieve head values
  report hFive.map = mf::getHeads(5);
  report hThree.map = mf::getHeads(3);
  report hOne.map = mf::getHeads(1);


  # retrieve drain leakage
  report dFive.map = mf::getDrain(5);
```

5.2. PCRaster Python model script

```
# This example script demonstrates the use of the
# PCRasterModflow extension. It recreates the example
# problem given in the user guide of MODFLOW-2000
# (Open file report 00-92)
```

```
#
# To run the script use
# python2.5 example.py

from PCRaster import *


setclone("clone.map")
mf = initialise(clone())

# grid specification
mf.createBottomLayer("bottom.map", "l1_top.map")
mf.addConfinedLayer("l2_top.map")
mf.addLayer("l3_top.map")
mf.addConfinedLayer("l4_top.map")
mf.addLayer("elev.map")

# simulation parameter
mf.setDISParameter(1, 0, 86400, 1, 1, 1)

# adding the boundary conditions to the MODFLOW model
mf.setBoundary("l1_bound.map", 1)
mf.setBoundary("l3_bound.map", 3)
mf.setBoundary("l5_bound.map", 5)

# adding the initial values to the MODFLOW model
mf.setInitialHead("head.map", 1)
mf.setInitialHead("head.map", 3)
mf.setInitialHead("head.map", 5)

# adding the conductivities to the MODFLOW model
mf.setConductivity(0, "hcond_l1.map", "vcond_l1.map", 1)
mf.setConductivity(0, "hcond_l1.map", "vcond_l2.map", 2)
mf.setConductivity(0, "hcond_l3.map", "vcond_l3.map", 3)
mf.setConductivity(0, "hcond_l3.map", "vcond_l4.map", 4)
mf.setConductivity(1, "hcond_l5.map", "vcond_l5.map", 5)

# solver package
mf.setSIP(50, 5, 1.0, 0.001, 0, 0.001)

mf.setDryHead(1E30)

# adding the drain
mf.setDrain("drain_elev.map", "drain_cond.map", 5)

# specifying the recharge
mf.setRecharge("rch.map", 1)

# adding well
mf.setWell("well_l1.map", 1)
mf.setWell("well_l3.map", 3)
mf.setWell("well_l5.map", 5)


# execute MODFLOW modelling engine
mf.run();

# retrieve head values
```

```
hFive = mf.getHeads(5)
report(hFive, "hFive.map")
hThree = mf.getHeads(3)
report(hThree, "hThree.map")
hOne = mf.getHeads(1)
report(hOne, "hOne.map")

# retrieve drain leakage
dFive = mf.getDrain(5);
report(dFive, "dFive.map")
```

Chapter 6. License

**** End-User License Agreement ****

This End-User License Agreement is a legal contract between you, as either an individual or a single business entity, and PCRaster Environmental Software (PES Consultancy BV).

1. Use of the software implies that you have read this license agreement and agree fully with what is stated in this agreement.
2. If you obtained the free Microsoft Windows version of PCRaster, for each personal registration you are licensed to use the software on one computer. For any other use of PCRaster you have purchased the correct license from PCRaster Environmental Software.
3. Any unauthorized duplication of PCRaster or its accompanying documentation is prohibited.
4. The software package PCRaster is developed by and copyright (c) since 1987 by PCRaster Environmental Software and the Faculty of Geographical Sciences of Utrecht University.
5. Use of the software is permitted under the following conditions:
   a. Not to sell or give away the software or any products which incorporate parts of or the total PCRaster package.
   b. Not to use the software or the practical exercises to give courses to third parties without prior approval of the owners of PCRaster.
   c. Not to develop any product which incorporate PCRaster or can be considered a shell executing PCRaster commands.
   d. Not to change the software, alter the software, decompile or use 'reverse engineering' or disassembly techniques, develop products based on the software or to create 'look and feel' products derived from the software.
   e. To accept the program as is, with no obligation on the part of the developers to provide corrections or assistance, beyond the support contract offered to you.
6. NO LIABILITY FOR CONSEQUENTIAL DAMAGES. In no event will the author be liable to you for any additional damages, including any lost profits, lost savings, or other incidental or consequential damages arising from the use of, or inability to use, this software and its accompanying documentation, even if the author has been advised of the possibility of such damages.
7. The authors of PCRaster make no warranty of any kind, expressed or implied, including without limitation any warranties of merchantability and/or fitness for a particular purpose. The authors do not assume any liability for the use of PCRaster beyond the original purchase price of this software.

**** End Of End-User License Agreement ****