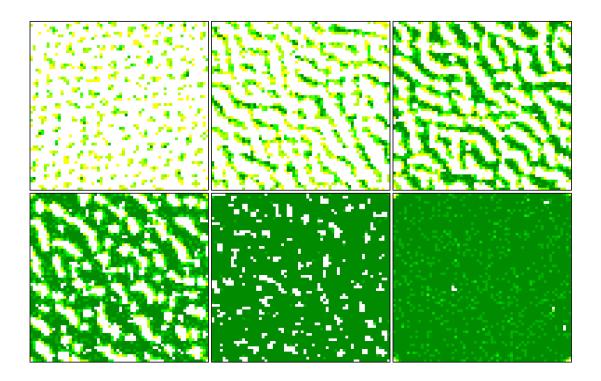
# **Ecohydraulical Feedback Simulation Documentation and Manual**



Gavan McGrath School of Earth and Environment, The University of Western Australia, M087, 35 Stirling Highway, Crawley, Western Australia, 6009

Christoph Hinz
Tobias Nanu Frechen
Brandenburg University of Technology, Hydrology and Water Resources Management,
Konrad-Wachsmann-Allee 6, 03046 Cottbus, Germany

DRAFT of 30<sup>th</sup> July, 2012, 16:45

## Contents

1	Intro	oduction		1		
2	The	heoretical principles 2				
	2.1	Simulation	n model	2		
	2.2		eneration	2		
	2.3	_	vater flow	2		
	2.4		ge facilitation	2		
	2.4		on and transpiration	2		
	_		•			
	2.6	-	n change	2		
	2.7	Microtop	ography	2		
3	Арр	lication		3		
	3.1	Build fro	n source	3		
		3.1.1 l	nix Systems	3		
	3.2	Installati	n	3		
		3.2.1	ystem requirements	3		
	3.3		on prerequisites	3		
	3.4		peration	3		
	-					
	3.5		rization	3		
	3.6		n	3		
	3.7			3		
	3.8	Appraise	and visualize results	3		
		3.8.1 i	R	3		
		3.8.2 i	R and LATEX (knitr-method)	3		
		3.8.3 i	Excel	3		
4	lmn	lementati	an			
4	-	lementati Overview		4		
4	4.1	Overview		<b>4</b>		
4	4.1 4.2	Overview Impleme	tation strategy	<b>4</b> 4		
4	4.1	Overview Impleme Subrouti	tation strategy	<b>4</b> 4 4		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1	tation strategy	<b>4</b> 4 4 4		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 6	tation strategy	<b>4</b> 4 4 4 4		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3	tation strategy es imulation reen-ampt infiltration anu: TwoDRandPos [cannot guess what this means]	<b>4</b> 4 4 4		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3	tation strategy	<b>4</b> 4 4 4 4		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3 \$ 4.3.4 \$ 4.3.4	tation strategy es imulation reen-ampt infiltration anu: TwoDRandPos [cannot guess what this means]	<b>4</b> 4 4 4 5		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3 \$ 4.3.4 \$ 4.3.5 \$ 6.5 \$	tation strategy es imulation reen-ampt infiltration revi <u>TwoDRandPos</u> [cannot guess what this means] inematic wave primer	<b>4</b> 4 4 4 5 5		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3 \$ 4.3.4 \$ 4.3.5 \$ 4.3.6 \$ 8.3.	tation strategy es imulation reen-ampt infiltration reven-ampt [cannot guess what this means] inematic wave primer inematic wave	<b>4</b> 4 4 4 5 5		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3 \$ 4.3.4 \$ 4.3.5 \$ 4.3.6 \$ 4.3.7 \$ 6	tation strategy es imulation reen-ampt infiltration TwoDRandPos [cannot guess what this means] inematic wave primer inematic wave inematic wave order neDRandList (a, mn)	44 44 44 45 55 66		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3 \$ 4.3.4 \$ 4.3.5 \$ 4.3.6 \$ 4.3.7 \$ 6.4.3.8 \$ 1.00 \$ 1	tation strategy es imulation reen-ampt infiltration rematic wave primer inematic wave inematic wave order neDRandList (a, mn) ookup for direction	44 44 44 45 55 66 67		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$\frac{9}{4.3.2} \$\frac{9}{4.3.3} \$\frac{1}{4} \text{4.3.5} \$\frac{1}{4} \text{4.3.6} \$\frac{1}{4} \text{4.3.8} \$\frac{1}{4} \text{4.3.8} \$\frac{1}{4} \text{4.3.9} \$\frac{1}{4} \text{4.3.9} \$\frac{1}{4} \text{4.3.9} \$\frac{1}{4} \text{4.3.9}	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration inematic wave primer inematic wave primer inematic wave order neDRandList (a, mn) ookup for direction outing with kernel	44 44 44 45 55 56 66 77		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$\frac{4}{3}.3.2 \$\frac{4}{4}.3.3 \$\frac{4}{4}.3.5 \$\frac{4}{4}.3.6 \$\frac{4}{4}.3.7 \$\frac{4}{3}.8 \$\frac{1}{4}.3.9 \$\frac{4}{4}.3.10 \$\frac{1}{4}.3.10 \$	tation strategy es imulation reen-ampt infiltration reinematic wave primer inematic wave inematic wave order inematic wave order neDRandList (a, mn) ookup for direction outing with kernel ifiltration Probability	44 44 44 45 55 66 67 77		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$\frac{9}{4.3.2} \$\frac{9}{4.3.3} \$\frac{1}{4.3.5} \$\frac{1}{4.3.6} \$\frac{1}{4.3.8} \$\frac{1}{4.3.8} \$\frac{1}{4.3.10} \$\frac{1}{4.3.11} \$\fr	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration inematic wave primer inematic wave order inematic wave order neDRandList (a, mn) ookup for direction outing with kernel ifiltration Probability ist Convolve	44 44 44 45 55 66 67 77 78		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$\frac{4}{3}.2 \$\frac{4}{3}.3 \$\frac{4}{3}.4 \$\frac{3}{3}.6 \$\frac{4}{3}.3 \$\frac{4}{3}.4 \$\frac{3}{3}.6 \$\frac{4}{3}.3 \$\frac{4}{3}.3 \$\frac{4}{3}.3 \$\frac{1}{3}.3 \$\frac{4}{3}.3 \$\frac{1}{3}.3 \$\fr	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration inematic wave primer inematic wave primer inematic wave order neDRandList(a,mn) ookup for direction outing with kernel ifiltration Probability ist Convolve rosion	44 44 44 45 55 66 67 77 78 88		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$\frac{4}{3}.3 \$\frac{4}{3}.4 \$\frac{1}{3}.5 \$\frac{1}{4}.3.5 \$\frac{1}{4}.3.6 \$\frac{1}{4}.3.7 \$\frac{1}{4}.3.8 \$\frac{1}{4}.3.10 \$\frac{1}{4}.3.11 \$\frac{1}{4}.3.12 \$\frac{1}{4}.3.13 \$\frac{1}{3}.43.13 \$\fra	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration rematic wave primer inematic wave primer inematic wave order neDRandList(a,mn) ookup for direction outing with kernel ifiltration Probability ist Convolve rosion plash	44 44 44 45 55 66 77 77 88 88		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 S 4.3.2 G 4.3.3 M 4.3.5 M 4.3.6 M 4.3.7 G 4.3.8 M 4.3.9 M 4.3.10 M 4.3.11 M 4.3.12 M 4.3.12 M 4.3.13 M 4.3.14 M 4.3	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration inematic wave primer inematic wave primer inematic wave order neDRandList(a, mn) ookup for direction outing with kernel ifiltration Probability ist Convolve rosion plash ind holes	44 44 44 45 55 66 67 77 88 88 89		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 S 4.3.2 G 4.3.3 M 4.3.5 M 4.3.6 M 4.3.7 G 4.3.8 M 4.3.9 M 4.3.10 M 4.3.11 M 4.3.12 M 4.3.12 M 4.3.13 M 4.3.14 M 4.3	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration rematic wave primer inematic wave primer inematic wave order neDRandList(a,mn) ookup for direction outing with kernel ifiltration Probability ist Convolve rosion plash	44 44 44 44 55 55 66 77 77 88 88		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3 \$ 4.3.4 \$ 4.3.5 \$ 4.3.6 \$ 4.3.7 \$ 4.3.8 \$ 4.3.10 \$ 4.3.11 \$ 4.3.12 \$ 4.3.13 \$ 4.3.14 \$ 4.3.15 \$ 6	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration inematic wave primer inematic wave primer inematic wave order neDRandList(a, mn) ookup for direction outing with kernel ifiltration Probability ist Convolve rosion plash ind holes	44 44 44 45 55 66 67 77 88 88 89		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$ 4.3.2 \$ 4.3.3 \$ 4.3.4 \$ 4.3.5 \$ 4.3.6 \$ 4.3.7 \$ 4.3.8 \$ 4.3.10 \$ 4.3.11 \$ 4.3.12 \$ 4.3.13 \$ 4.3.14 \$ 4.3.15 \$ 6 4.3.16 \$ 1.00 \$ 1	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration inematic wave primer inematic wave primer inematic wave order neDRandList(a, mn) ookup for direction outing with kernel ifiltration Probability ist Convolve rosion plash ind holes vaporation	44 44 44 44 55 55 66 67 77 78 88 88 99		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 S 4.3.2 G 4.3.3 M 4.3.4 M 4.3.5 M 4.3.7 G 4.3.8 M 4.3.11 M 4.3.12 M 4.3.12 M 4.3.13 M 4.3.14 M 4.3.15 M 4.3.16 M 4.3.17 M 4.3.16 M 4.3.17 M 4.	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration rematic wave primer inematic wave primer inematic wave order neDRandList(a,mn) ookup for direction outing with kernel ifiltration Probability ist Convolve rosion plash ind holes vaporation eighbours	44 44 44 44 55 55 66 77 77 88 88 91 10		
4	4.1 4.2	Overview Impleme Subrouti 4.3.1 \$\frac{4}{3}.3 \$\frac{4}{3}.4 \$\frac{3}{3}.5 \$\frac{4}{3}.6 \$\frac{4}{3}.7 \$\frac{4}{3}.10 \$\frac{1}{4}.3.12 \$\frac{1}{4}.3.13 \$\frac{1}{3}.43.14 \$\frac{1}{4}.3.15 \$\frac{1}{4}.3.17 \$\frac{1}{4}.3.18 \$\frac{1}{4}.3	tation strategy es imulation reen-ampt infiltration reen-ampt infiltration  rematic TwoDRandPos [cannot guess what this means] inematic wave primer inematic wave order neDRandList (a, mn) ookup for direction outing with kernel ifiltration Probability ist Convolve rosion plash ind holes vaporation eighbours SDS	44 44 44 44 55 55 66 67 77 77 88 88 91 10 10		

		4.3.21	New GD8 flow directions	12
		4.3.22	fdirLookup(dirnxy, idirn)	12
		4.3.23	Array sort	12
		4.3.24	endShift	13
		4.3.25	makeOrds	13
		4.3.26	Vegetation Change	13
	4.4	Custon	nizability	14
List	t of F	igures,	Tables and Listings	15
Ind	ex			18
Ref	feren	ces		20
Α	Арре	endix		21

#### **Draft status**

- Suggestions for overall structuring welcomed!
- what means Dimensions: [:,:] ?
- subroutines and comments on the variables are from file <code>coupledModel.f90</code>
- \subroutine{<name>} {<arguments>} for displaying a subroutine with its arguments. <name> automatically gets an (emphasized) index entry and a label.
- \inout{<input entries>} {<output entries>} is for a list of input and output arguments. Missing values get replaced by "no input" or "no output".
- \inoutentry[<option>] {<variableName>}{type}{dimensions}{description} for an entry in the input output list. <option> can be empty or "emph" for an emphasized index entry or "none" for no index entry.
- \begin{usessubs} is an environment of used subroutines.
- \usessubentry{<subroutineName>} is for an entry in the usessubs environment. Entries automatically get an (not emphasized) index entry and a reference to the associated subroutine.
- I am using the keyword index to index all parameter and variable names. Good idea? Or shall we make a list with all parameter and variable names and their description and unit? Like:

Table 1: variables variable type dimensions nom description [8-byte real] [4] climate parameters climParams [8-byte real] [6] vegetation and soil kernel parameters infiltParams [8-byte real] [8] evaporation parameters evapParams  $[\mathrm{m}^3/\mathrm{S}]$ discharge [integer] [m,n]Qdischarge alpha [8-byte real] [m,n]

My suggestion for collaboration: let's use the package "trackchanges". Nanu: Editing [could look like this] with the following commands:

```
\add[editor]{added text}
\remove[editor]{removed text}
\change[editor]{removed text}{added text}
\note[editor]{note text}
\annote[editor]{text to annotate}{note text}
```

# 1 Introduction

#### 2 Theoretical principles

#### 2.1 Simulation model

Runoff, soil moisture storage, transpiration and plant-bare soil transitions are numerically modeled on a lattice of square cells. The model simulates the following spatially distributed water balance:

$$\frac{\partial w_i}{\partial t} = P_i + R_i - Q_i - E_i - \sum_n T_n \tag{1}$$

- 2.2 **Runoff generation**
- 2.3 Surface water flow
- 2.4 Short range facilitation
- 2.5 **Evaporation and transpiration**
- 2.6 **Vegetation change**
- 2.7 Microtopography

## 3 Application

#### 3.1 Build from source

#### 3.1.1 Unix Systems

To use the gfortran compiler execute the following in terminal:

```
gfortran coupledModel.f90
```

On Unix systems an .out file is generated. To execute this, type the following into the terminal:

./a.out

#### 3.2 Installation

- 3.2.1 System requirements
- 3.3 Application prerequisites
- 3.4 Data preperation
- 3.5 Parameterization
- 3.6 Simulation
- 3.7 Output
- 3.8 Appraise and visualize results
- 3.8.1 in R
- 3.8.2 in R and LTEX (knitr-method)

```
http://yihui.name/knitr/
```

#### 3.8.3 in Excel

## 4 Implementation

#### 4.1 Overview

#### 4.2 Implementation strategy

#### 4.3 Subroutines

#### 4.3.1 Simulation

```
SimCODE(m, n, mn, simflags, climParams, infiltParams, evapParams,
vegParams, erParams, resultsFID)
```

#### input

m	[integer]		number of rows
n	[integer]		number of colums
mn	[integer]		$= m \cdot n$
simflags	[integer]	[6]	[1] $\dots$ is for this, [2] $\dots$ is for that, Nanu: [should be explained!]
climParams	[8-byte real]	[4]	Nanu: climate parameters [how to input these?]
infiltParams	[8-byte real]	[6]	Nanu: vegetation and soil kernel parameters
evapParams	[8-byte real]	[8]	evaporation parameters
vegParams	[8-byte real]	[5]	vegetation Nanu: and soil kernel parameters
erParams	[8-byte real]	[4]	Nanu: vegetation and soil kernel parameters [what specific?]
resultsFID	[character]	[len=8]	results file id code

#### output

no output

#### Uses subroutines:

```
GD8(), cf. page 11
InfiltProb(), cf. page 7
RoutingWithKernel(), cf. page 7
Erosion(), cf. page 8
Splash(), cf. page 8
Evaporation(), cf. page 10
VegChange(), cf. page 13
```

#### 4.3.2 Green-ampt infiltration

```
GAInfilt(m, n, dt, inflow, Ksat, wfs, cumInfilt, infex)
```

Nanu: Green-ampt infiltration calculation. [Literature?]

#### input

```
m, n [integer]
dt [8-byte real]
inflow [8-byte real] [m,n] surface runon + precipitation (R+P)
wfs [8-byte real] [m,n] Nanu: wetting front suction * moisture deficit [formula?]
cumInfilt [8-byte real] [m,n] Nanu: cumulative infiltration [nomenclature?]
```

#### output

```
infex [8-byte real] [m,n] infiltration excess
cumInfilt [8-byte real] [m,n] cumulative infiltration
```

#### 4.3.3 Nanu: TwoDRandPos [cannot guess what this means]

```
input
  m, n     [integer]
  randOrder [integer] [mn, 2]

output
  randOrder [integer] [mn, 2]

Uses subroutines:
  OneDRandList(), cf. page 6
```

#### 4.3.4 Kinematic wave primer

```
KWPrimer(m, n, topog, manningsN, mask, solOrder, flowdirns, alpha, deltax,
solMax)
```

Iterate over space to generate calculation a mask. The mask is a  $9 \times 1$  array where 1s indicate that a neighbour (as defined by the position in mask(x,y)) discharges to the cell at x,y

#### input

```
m,n [integer]
topog [integer] [mn,n]
manningsN [integer] [mn,n]
```

#### output

```
deltax [8-byte real] [m,n]
alpha [8-byte real] [m,n]
solMax [integer]
solOrder [integer] [m,n]
flowdirns [integer] [m,n] flow directions
mask [integer] [m,n,9]
```

#### Uses subroutines:

```
GD8(), cf. page 11
KMWOrder(), cf. page 6
Neighbours(), cf. page 10
Lookupfdir(), cf. page 7
```

#### 4.3.5 Kinematic wave

```
KinematicWave(m, n, ndx, dt, iex, flowdirns, solOrder, solMax, mask, alpha,
deltax, disOld, disNew)
```

This subroutine calculates the kinematic wave equation for surface runoff in a network for a single time step. The flow network is given by the GD8 algorithm calculated on the topography. We assume no interaction between adjacent flow pathways i.e. water does not overflow in a direction not specified by the GD8 network. iex is the excess water from Nanu: precip - infiltration + GW discharge [display as formula?].

$$iex(m, n, 1) = qit$$

Nanu: [should this be displayed as formula or as code?]

$$iex(m, n, 2) = qitPlus$$

Nanu: manningsN the roughness coefficient [is this in the right place?]

Nanu: init a value passed to initialise the variables [same for this]

#### input

m, n	[integer]		
ndx	[integer]		
solMax	[integer]		
flowdirns	[integer]	[m,n]	flow directions
solOrder	[integer]	[m,n]	
mask	[integer]	[m,n]	
alpha	[8-byte real]	[m,n]	
deltax	[8-byte real]	[m,n]	
dt	[8-byte real]		time step
disOld,disNew	[8-byte real]	[m,n,ndx]	
iex	[8-byte real]	[m,n,2]	

#### output

disOld, disNew [8-byte real] [m,n,ndx]

#### 4.3.6 Kinematic wave order

```
KMWOrder(flowdirns, m, n, solutionOrder)
```

Subroutine assigns values to the matrix solutionOrder to tell the Kinematic Wave subroutine in which order to solve the KM equation on the drainage network. Values of 1 assigned to top of catchment, 2 to 1st downstream node etc.. Value at a point is the maximum travel distance to that point from all points above it.

#### input

```
m,n [integer]
flowdirns [integer] [m,n] flow directions
```

#### output

```
solutionOrder [integer] [n,n]
```

#### Uses subroutines:

Lookupfdir(), cf. page 7

#### 4.3.7 OneDRandList(a,mn)

```
OneDRandList(a, mn)
```

#### input

```
mn [integer]
a [integer] [mn,2]
```

#### output

a [integer] [mn,2]

#### 4.3.8 Lookup for direction

```
input
  dirn [integer]

output
  dx, dy [integer]
```

#### 4.3.9 Routing with kernel

```
RoutingWithKernel(m, n, mn, precip, infiltKern, storeKern, newflowdirns,
topog, store, discharge, outflow)
```

#### input

```
[integer]
m, n
                 [integer]
mn
infiltKern
                 [8-byte real] [m,n]
                 [8-byte real] [m,n]
storeKern
                 [8-byte real] [m,n]
topog
precip
                 [integer]
                              [m,n]
store
                 [integer]
                              [m,n]
newflowdirns [integer]
                              [m,n] flow directions
```

#### output

```
outflow [integer]
store [integer] [m,n]
newflowdirns [integer] [m,n] new flow directions
discharge [integer] [m,n]
```

**Notice:** Periodic boundary conditions can only really be defined simply for an inclined plane with two adjacent edges defined as the boundary from which particles are routed to the opposite boundary. For simplicity it is assumed that the landscape slopes downwards in the direction of lower x and y.

#### Uses subroutines:

```
OneDRandList(), cf. page 6
TwoDRandPos(), cf. page 5
Lookupfdir(), cf. page 7
Neighbours(), cf. page 10
fdirLookup(), cf. page 12
```

#### 4.3.10 Infiltration Probability

```
InfiltProb(veg, m, n, K0, ie, rfx, rfy, kf, Kmax, dx, dy, iProb)
```

This subroutine calculates the spatial distributed infiltration probability.

```
input
```

[integer] dimensions of the spatial arrays m,n rfx, rfy [integer] maximum radius for plant effects on soil properties kf [integer] [integer] [m,n] vegetation matrix veg K0 [8-byte real] [8-byte real] ie [8-byte real] Kmax dx, dy [8-byte real] length scales of lattice

#### output

iProb [8-byte real] [m,n] infiltration probability matrix

#### 4.3.11 List Convolve

ListConvolve (base, kernel, convol, m, n, m1, n1)

#### input

m,n [integer]
m1,n1 [integer]
base [integer] [m,n]
kernel [integer] [m,n]

#### output

convol [ [m,n]

#### 4.3.12 **Erosion**

Erosion(discharge, topog, newflowdirns, flowResistance, m, n)

#### input

m,n [integer] dimensions of the spatial arrays discharge [integer] [m,n] discharge has units of  $^{mm}/_{year}$  newflowdirns [integer] [m,n] new flow directions

flowResistance [8-byte real] [m,n] flow resistance; effective  $d_{40}$  grainsize in mm

topog [8-byte real] [m,n] topography

#### output

topog [8-byte real] [m,n] topography

#### Uses subroutines:

Lookupfdir(), cf. page 7

#### 4.3.13 Splash

```
Splash (topog, veg, Dv, Db, m, n)
```

#### 4.3.14 Find holes

```
input
  newtopog [8-byte real] [:,:] topography

output
  holes [8-byte real] [:,:]
```

Listing 1: a program listing could look like this; notice the language sensitive formatting

```
SUBROUTINE FindHoles (newtopog, holes)
2
   IMPLICIT NONE
3
   REAL*8, DIMENSION(:,:), INTENT(IN) :: newtopog
4
5
   INTEGER, INTENT(OUT) :: holes
6
7
   INTEGER :: m,n,i,j,k,l
   m=SIZE (newtopog, 1)
8
9
   n=SIZE (newtopog, 2)
10
   DO i=2, m-1
11
   DO j=2, n-1
12
13
        holes=0
14
        DO k=-1, 1
        DO 1=-1,1
15
             IF (\text{newtopog}(i,j) < \text{newtopog}(i + k, j + l)) THEN
16
17
                 holes = holes + 1
18
            END IF
19
        END DO
20
        END DO
21
        IF (holes.ge.8) THEN
            holes = 1
22
            RETURN
23
24
        END IF
25
   END DO
26
   END DO
27
   END SUBROUTINE FindHoles
28
```

Listing 1 shows, how code can be inserted into the document. With line numbers and Fortran specific code highlighting.

It is possible to refer to individual lines inside the listing with  $\triangle T_{EX}$  commands  $\label{}$  and  $\ref{}$ , so references get updated if the code changes. For example line 16 in listing 1, where the first IF-condition begins.

The code in the listing could be loaded from an external file (for example the actual Fortran file).

#### 4.3.15 Evaporation

```
Evaporation(veg, eTActual, bareE, store, tsteps, rcx, rcy, kc, dx, dy, params)
```

This version cycles through sites and evaporates water from site and neighbouring sites if vegetated.

#### input

veg		[integer]	[:,:]
eTActual		[integer]	[:,:]
bareE		[integer]	[:,:]
store		[integer]	[:,:]
tsteps		[integer]	
rcx, rcy,	kc	[8-byte real]	
dx, dy		[8-byte real]	
params		[8-byte real]	[7]

#### output

store	[integer]	[:,:]
eTActual	[integer]	[:,:]
bareE	[integer]	[:.:]

#### Uses subroutines:

TwoDRandPos(), cf. page 5

#### 4.3.16 Neighbours

```
Neighbours (order, posij, dom, neighbs)
```

#### input

```
order [integer]
posij [integer] [2]
dom [integer] [2]
```

#### output

```
neighbs [integer] [Nanu: (order*2+1)**2,2 [as formula?]]
```

#### 4.3.17 LSDs

```
LSDs (order, posxy, topog, m, n, lsdList)
```

This function returns the matrix positions:

LSD1: the position of the neighbouring cell with the steepest slope downhill

LSD2: the position of the neighbouring cell with second steepest slope adjacent LSD1

otherpos: the position of the other neighbouring cell the mirror reflection about LSD1 of LSD2

```
input
```

```
m, n [integer]
posxy [integer] [2]
topog [8-byte real] [m,n]
```

#### output

```
lsdList [integer] [3,2]
```

#### Uses subroutines:

```
Neighbours(), cf. page 10
RotateArray(), cf. page 11
```

#### 4.3.18 Array rotation

```
RotateArray(list, m, n, leftorRight)
```

#### input

```
m,n [integer]
list [integer] [m,n]
leftorRight [integer]
```

#### output

list [integer] [m,n]

#### 4.3.19 Pos1D

```
Posld(list, m, n, match, rownum)
```

#### input

```
m,n [integer]
list [integer] [m,n]
match [integer] [n]
```

#### output

rownum [integer] [m]

#### 4.3.20 GD8 fow directions

```
GD8 (topog, flowdirns, m, n)
```

#### input

m, n [integer]

topog [8-byte real] [m,n]

#### output

flowdirns [integer] [m,n] flow directions

```
Uses subroutines:
```

```
makeOrds(), cf. page 13
Posld(), cf. page 11
endShift(), cf. page 13
LSDs(), cf. page 10
fdirLookup(), cf. page 12
RotateArray(), cf. page 11
```

#### 4.3.21 New GD8 flow directions

```
NewGD8 (topog, lakes, flowdirns, m, n)
```

Calculates flow directions following the GD8-algorithm of Paik (2008)

#### input

```
m,n [integer]
lakes [integer] [m,n]
topog [8-byte real] [m,n]
```

#### output

flowdirns [integer] [m,n] flow directions

#### Uses subroutines:

```
makeOrds(), cf. page 13
Posld(), cf. page 11
endShift(), cf. page 13
LSDs(), cf. page 10
fdirLookup(), cf. page 12
Neighbours(), cf. page 10
```

#### 4.3.22 fdirLookup(dirnxy, idirn)

```
fdirLookup(dirnxy, idirn)
input
  dirnxy [integer] [2]
output
  idirn [integer]
```

#### 4.3.23 Array sort

```
qsortd(x, ind, n, incdec)
```

This subroutine uses an order  $n \cdot \log(n)$  quick sort to sort a real (double precision/8-byte) array x(n) into increasing order.

#### input

```
x(n) [8-byte real] vector of length n to be sorted n [integer] length of the array x(n) incdec [integer] if positive the ind is returned so values decreasing order incdec [integer]
```

#### output

```
ind (n) [integer] vector of length \geq n; sequence of indices 1, \ldots, n permuted in the same fashion as x would be: y(i) = x \, (ind(i))
```

ind is initialized to the ordered sequence of indices  $1, \ldots, n$ , and all interchanges are applied to ind. x is devided into two portions by picking a central element t. The first and last elements are compared with t, and interchanges are applied as necessary so that the three values are in ascending order. Interchanges are then applied so that all elements greater than t are in the upper portion of the array and all elements less than t are in the lower portion. The upper and lower indices of one of the portions are saved in local arrays, and the process is repeated iteratively on the other portion. When a portion is completely sorted, the process begins again by retrieving the indices bounding another unsorted portion.

Note: IU and IL must be dimensioned  $\geq \log(n)$  where  $\log$  has base 2.

Credit goes to Robert Renka Oak Ridge Natl. Lab.

#### 4.3.24 endShift

```
input
  rown [integer]
  n   [integer]
  mn  [integer]
  arr [integer] [mn,n]

output
  arr [integer] [mn,n]
```

#### 4.3.25 makeOrds

```
makeOrds (topog, ords, m, n)
input
   m,n [integer]
   topog [8-byte real] [m,n]

output
   ords [integer] [size(topog),2]

Uses subroutines:
   qsortd(), cf. page 12
```

#### 4.3.26 Vegetation Change

```
VegChange (veg, m, n, vegParams, store, actualET, isEmerge, isGrow)
```

#### input

m,n [integer] isGrow [integer] [integer] isEmerge vegParams [8-byte real] [5] veg [integer] [m,n]store [integer] [m,n]actualET [integer] [m,n]

#### output

veg[integer][m,n]store[integer][m,n]actualET[integer][m,n]

## 4.4 Customizability

# List of Figures

## **List of Tables**

# List of Listings

# Index

a, 6	v∩ Q
•	K0, 8
actualET, 14	kernel, 8
alpha, 5, 6	kf, 8
arr, 13	kinematic wave, 5
l 10	KinematicWave(), <b>5</b>
bareE, 10	Kmax, 8
base, 8	KMWOrder(), 5, <b>6</b>
	KWPrimer(), <b>5</b>
climParams, 4	1-1 10
convol, 8	lakes, 12
cumInfilt, 4	leftorRight, 11
5.6	list, 11
deltax, 5, 6	ListConvolve(), 8
dirn, 7	Lookupfdir(), 5, 6, <b>7</b> , 7, 8
dirnxy, 12	lsdList, 11
discharge, 7, 8	LSDs (), <b>10</b> , 12
disOld, disNew, 6	4
disOld,disNew,6	m, <b>4</b>
dom, 10	m, n, 4-9, 11-14
dt, 4, 6	m1, n1, 8
Dv, Db, 9	makeOrds(), 12, <b>13</b>
dx, dy, 8, 10	manningsN, 5
dx <b>,</b> dy, 7	mask, 5, 6
	match, 11
endShift(), 12, <b>13</b>	mn, 4, 6, 7, 13
Erosion(), 4, <b>8</b>	4 10
erosion, 8	n, 4, 13
erParams, 4	ndx, 6
eTActual, 10	Neighbours (), 5, 7, 9, <b>10</b> , 11, 12
Evaporation(), 4, ${f 10}$	neighbs, 10
evaporation, 10	newflowdirns, 7, 8
evapParams, 4	NewGD8(), <b>12</b>
	newtopog, 9
fdirLookup(),7, <b>12</b> ,12	0DD11'()
FindHoles(), $9$	OneDRandList(), 5, <b>6</b> , 7
flow directions, 12	order, 10
flowdirns, 5, 6, 11, 12	ords, 13
flowResistance, 8	outflow, 7
	params, 10
GAInfilt(), <b>4</b>	Pos1d(), <b>11</b> , 12
GD8(), 4, 5, <b>11</b>	posij, 10
GD8-algorithm, 12	posry, 11
	precip, 7
holes, 9	precip, /
	qsortd(), <b>12</b> , 13
idirn, 12	950164(), 12, 10
ie, 8	randOrder, 5
iex, 6	rcx, rcy, kc, 10
incdec, 13	resultsFID, 4
ind(n), 13	rfx, rfy, 8
infex, 4	RotateArray(), <b>11</b> , 11, 12
infiltKern,7	RoutingWithKernel(), 4, 7
infiltParams, 4	rown, 13
<pre>InfiltProb(), 4, 7</pre>	rownum, 11
infiltration, 7	
inflow, 4	SimCODE(), 4
iProb, 8	simflags, 4
isEmerge, 14	simulation, 4
isGrow, 14	solMax, 5, 6

solOrder, 5, 6
solutionOrder, 6
Splash(), 4, 8
store, 7, 10, 14
storeKern, 7

topog, 5, 7-9, 11-13 tsteps, 10 TwoDRandPos(), **5**, 7, 10

veg, 8-10, 14 VegChange(), 4, **13** vegParams, 4, 14

water balance, 2 wfs, 4

x(n), 13

## References

Paik, K. (2008). Global search algorithm for nondispersive flow path extraction. *Journal of Geophysical Research*, 113(F4):F04001.

# A Appendix