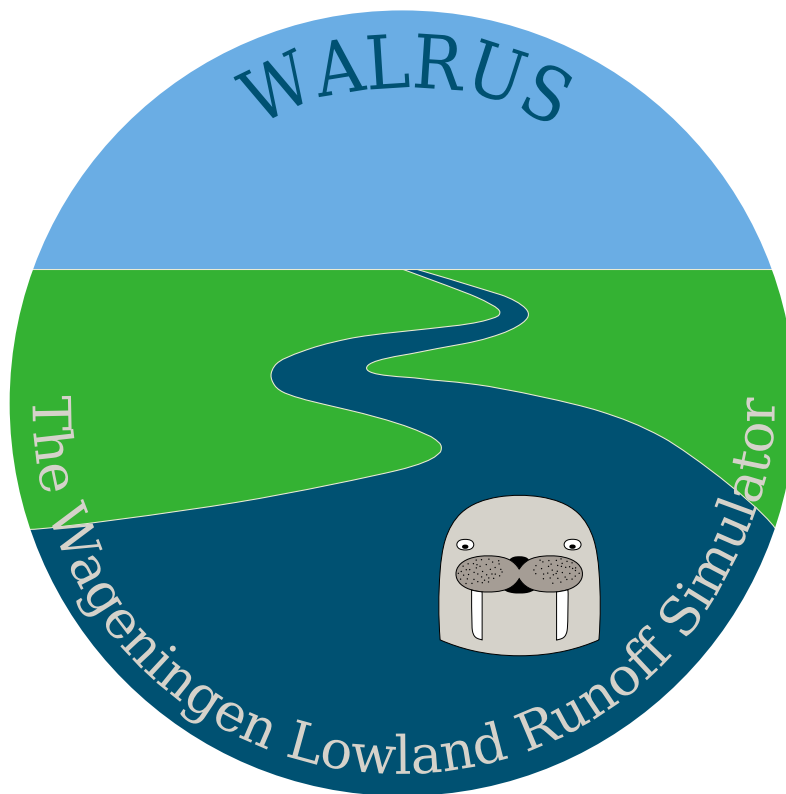


# The Wageningen Lowland Runoff Simulator WALRUS 1.09

## User manual



Claudia Brauer, Paul Torfs, Ryan Teuling and Remko Uijlenhoet  
Hydrology and Quantitative Water Management Group  
Wageningen University, The Netherlands

25 May 2016



WAGENINGEN UNIVERSITY  
WAGENINGENUR

**Authors**

Dr. ir. C.C. Brauer

Drs. P.J.J.F. Torfs

Dr. ir. A.J. Teuling

Prof. dr. ir. R. Uijlenhoet

Hydrology and Quantitative Water Management Group

Wageningen University

The Netherlands

[www.wageningenur.nl/hwm](http://www.wageningenur.nl/hwm)

**E-mail contact**

[Claudia.Brauer@wur.nl](mailto:Claudia.Brauer@wur.nl)

**WALRUS downloads**

[www.github.com/ClaudiaBrauer/WALRUS](https://www.github.com/ClaudiaBrauer/WALRUS)

# Contents

<b>1</b>	<b>Introduction / preface</b>	<b>1</b>
<b>2</b>	<b>Model description</b>	<b>3</b>
2.1	General overview	3
2.2	Precipitation and wetness index	5
2.3	Evapotranspiration	5
2.4	Storage deficit	6
2.5	Equilibrium storage deficit	6
2.6	Percolation and capillary rise	7
2.7	Groundwater	7
2.8	Quickflow	8
2.9	Surface water	8
2.10	Seepage and surface water supply	9
2.11	Large-scale ponding and flooding	9
2.12	Outlook: possible model extensions	9
<b>3</b>	<b>Model implementation</b>	<b>11</b>
3.1	Code set-up	11
3.2	Initial conditions	11
3.3	Parameters	11
3.4	Forcing	11
3.5	If-statements	11
3.6	Integration scheme	12
3.7	Time step	12
3.8	Water balance	13
<b>4</b>	<b>The WALRUS R-package</b>	<b>15</b>
4.1	R and RStudio	15
4.2	Core functions	15
4.3	Preprocessing	15
4.4	Default functions	15
4.5	Postprocessing	16
4.6	Graphical user interface	17
<b>5</b>	<b>Building your own WALRUS</b>	<b>19</b>
5.1	Suitability	19
5.2	Installing the package	19
5.3	Help	20
5.4	File organization	20
5.5	Data requirements	20
5.6	Preprocessing	21
5.7	Calibration	21
5.8	Running and postprocessing	21
<b>6</b>	<b>Examples</b>	<b>23</b>
6.1	Basic example	23
6.2	Changing default functions	24
6.3	Changing parameters and initial conditions	24
6.4	Calibration	26
<b>7</b>	<b>FAQ</b>	<b>33</b>
7.1	Things to check during calibration	33
7.2	Questions	33
7.3	Error messages	33
	<b>Bibliography</b>	<b>35</b>
	<b>Index</b>	<b>37</b>



# 1 | Introduction / preface

The Wageningen Lowland Runoff Simulator (WALRUS) is a lumped rainfall-runoff model for catchments with shallow groundwater. It was developed in 2013 at the Hydrology and Quantitative Water Management Group of Wageningen University to fill the gap between complex, spatially distributed models which are often used in lowland areas and simple, parametric (conceptual) models which have mostly been developed for mountainous catchments. WALRUS is lumped (spatially non-distributed), mass balance conserving, fast, programmed in R and freely downloadable.

The model is described in detail in two peer reviewed papers which have been published in open access journals (Brauer et al., 2014a,b). We refer to the paper in Geoscientific Model Development for discussions about the need for simple rainfall-runoff models, a review on lowland-specific hydrological processes, and elaborate explanations of the model structure and code (Brauer et al., 2014a). We refer to the paper in Hydrology and Earth System Sciences for examples in freely draining and polder catchments, as well as for sensitivity analyses and uncertainty analyses (Brauer et al., 2014b). Please cite these papers when you use the model.

In addition to the papers, we wrote this user manual to facilitate application. The user manual is still “work in progress” (especially the frequently asked questions in Chapter 7) and new versions with more examples and improved instructions will appear from time to time. This also holds for the model (and the R package) itself. We welcome advice and examples from other users to improve this manual and the model.

We realise that not everyone will read this manual from beginning to end. In Chapter 2 we explain the model set-up and in Chapter 3 we explain the basics of the model code. Chapters 2 and 3 are largely copied from Brauer et al. (2014a), but contain a little more explanation about the model relations and fewer examples (field observations and numerical experiments). Reading these chapters will teach you which variables and parameters the model contains, which functions are implemented and how model variables can be compared to field observations.

If you want to get started as quickly as possible, you can fast forward to Chapter 4, where we describe the R-package and model functions and Chapter 5, where we describe the steps you have to take to apply WALRUS to your own catchment. In Chapter 6 we show some examples, which you can use as a starting point. Frequently asked questions (Chapter 7) and an index in the back can help you when you encounter problems.

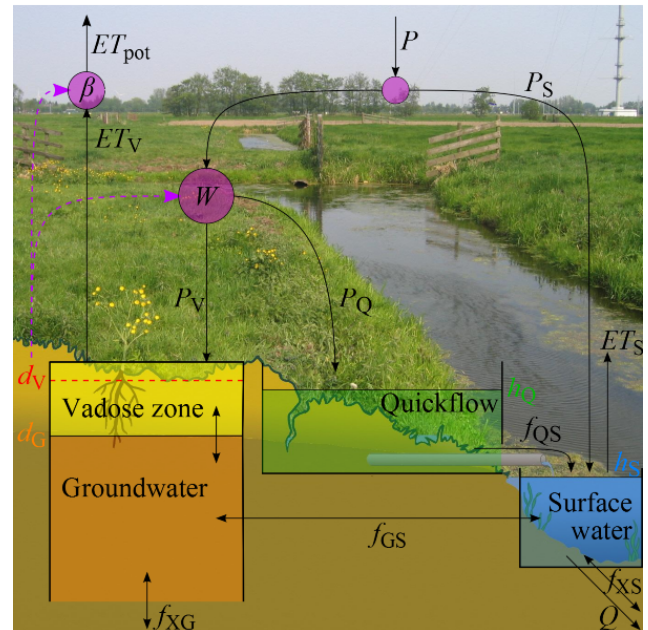


Figure 1.1: The model structure and its physical representation.



## 2 | Model description

In this Chapter we provide a detailed description of all model components: reservoirs, states, fluxes, parameters and feedback mechanisms. There are several relations between model variables in the model which can be specified by the user. We implemented defaults for these relations, such that WALRUS can be used directly by practitioners, while retaining the option to change them for research purposes. This Chapter is largely based on the Section 5 of the paper in GMD (Brauer et al., 2014a). If you are anxious to get WALRUS running and want to learn about what the fluxes actually mean later, you can decide to skip this Chapter for now and return to it later.

### 2.1 General overview

WALRUS is a water balance model with three reservoirs and fluxes between the reservoirs. The model can be split into 5 compartments (Fig. 2.1, for abbreviations of variables, see Tab. 2.1):

1. *Land surface* – at the land surface, water is added to the different reservoirs by precipitation ( $P$ ). A fixed fraction is led to the surface water reservoir ( $P_S$ ). The soil wetness index ( $W$ ) determines which fraction of the remaining precipitation percolates slowly through the soil matrix ( $P_V$ ) and which fraction flows towards the surface water via quick flow routes ( $P_Q$ ). Water is removed by evapotranspiration from the vadose zone ( $ET_V$ ) and surface water reservoir ( $ET_S$ ).
2. *Vadose zone within the soil reservoir* – the vadose zone is the upper part of the soil reservoir and extends from the soil surface to the dynamic groundwater table ( $d_G$ ), including the capillary fringe. The dryness of the vadose zone is characterised by a single state: the storage deficit ( $d_V$ ), which represents the effective volume of empty pores per unit area. It controls the evapotranspiration reduction ( $\beta$ ) and the wetness index ( $W$ ).
3. *Groundwater zone within the soil reservoir* – the phreatic groundwater extends from the groundwater depth ( $d_G$ ) downwards, thereby assuming that there is no shallow impermeable soil layer and allowing groundwater to drop below the depth of the drainage channels ( $c_D$ ) in dry periods. The groundwater table responds to changes in the unsaturated zone storage and determines together with the surface water level groundwater drainage or infiltration of surface water ( $f_{GS}$ ).
4. *Quickflow reservoir* – all water that does not flow through the soil matrix, passes through the quickflow reservoir to the surface water ( $f_{QS}$ ). This represents macropore flow through drainpipes, animal burrows

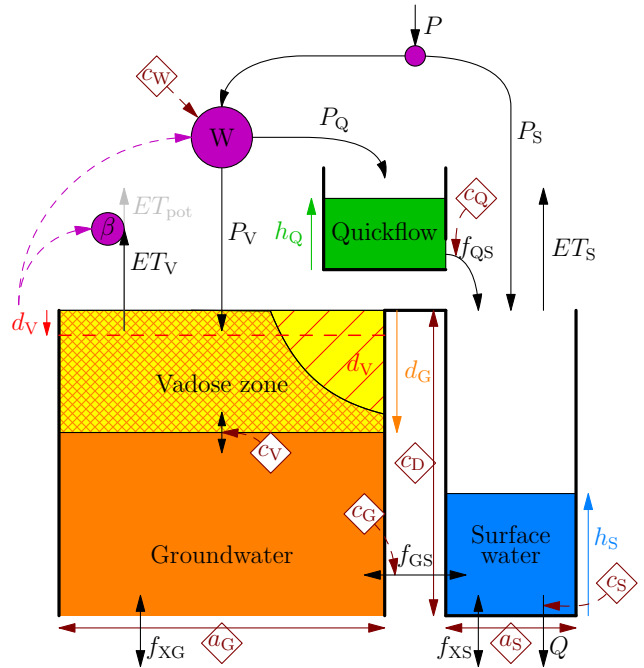


Figure 2.1: Overview of the model structure with the five compartments: land surface (purple), vadose zone within the soil reservoir (yellow/red hatched), groundwater zone within the soil reservoir (orange), quickflow reservoir (green) and surface water reservoir (blue). Fluxes are black arrows, model parameters brown diamonds and states in the colour of the reservoir they belong to. For a complete description of all variables, see Table 2.1 and Sec. 2.1. The names of the fluxes are derived from the reservoirs (for example  $f_{XS}$ :  $f$  stands for flow, the  $X$  for external and the  $S$  for surface water – water flowing from outside the catchment into the surface water network).

and soil cracks, but also local ponding and overland flow.

5. *Surface water reservoir* – the surface water reservoir has a lower boundary (the channel bottom  $c_D$ ), but no upper boundary. Discharge ( $Q$ ) is computed from the surface water level ( $h_S$ ).
6. *External fluxes* – water can be added to or removed from the soil reservoir by seepage ( $f_{XG}$ ) and to/from the surface water reservoir by surface water supply or extraction ( $f_{XS}$ ).

The area of the surface water reservoir  $a_S$  is the fraction of the catchment covered by ditches and channels, which is supplied by the user and can generally be derived from maps. The area of the soil reservoir  $a_G$  is the remainder ( $1 - a_S$ ). The area of the quickflow reservoir is taken equal to  $a_G$ , but this is arbitrary since the outflow depends on the volume of water in the reservoir and a

Table 2.1: Overview of variables, parameters and functions. All fluxes are catchment averages, both external ones (including  $Q$  and  $f_{XS}$ ) and internal fluxes (which are multiplied with the relative surface area of the reservoir in question). Note that  $d_V$ ,  $h_Q$  and  $h_S$  result from the mass balances in the three reservoirs, while  $d_G$  is only used as pressure head to compute the groundwater drainage flux. The names of the fluxes are derived from the reservoirs (for example  $f_{XS}$ :  $f$  stands for flow, the  $X$  for external and the  $S$  for surface water – water flowing from outside the catchment into the surface water network).

<b>States</b>			
$d_V$	storage deficit	$\rightarrow \frac{dd_V}{dt} = -\frac{f_{XG} + P_V - ET_V - f_{GS}}{a_G}$	[mm]
$d_G$	groundwater depth	$\rightarrow \frac{dd_G}{dt} = \frac{d_V - d_{V,eq}}{c_V}$	[mm]
$h_Q$	level quickflow reservoir	$\rightarrow \frac{dh_Q}{dt} = \frac{P_Q - f_{QS}}{a_Q}$	[mm]
$h_S$	surface water level	$\rightarrow \frac{dh_S}{dt} = \frac{f_{XS} + P_S - ET_S + f_{GS} + f_{QS} - Q}{a_S}$	[mm]
<b>Dependent variables</b>			
$W$	wetness index	$= \text{func}(d_V)$	[-]
$\beta$	evapotranspiration reduction factor	$= \text{func}(d_V)$	[-]
$d_{V,eq}$	equilibrium storage deficit	$= \text{func}(d_G)$	[mm]
<b>External fluxes: input</b>			
$P$	precipitation		[mm h <sup>-1</sup> ]
$ET_{\text{pot}}$	potential evapotranspiration		[mm h <sup>-1</sup> ]
$Q_{\text{obs}}$	discharge (for calibration and $Q_0$ )		[mm h <sup>-1</sup> ]
$f_{XG}$	seepage (up/down)/extraction	[mm h <sup>-1</sup> ]	
$f_{XS}$	surface water supply/extraction		[mm h <sup>-1</sup> ]
<b>External fluxes: output</b>			
$ET_{\text{act}}$	actual evapotranspiration	$= ET_V + ET_S$	[mm h <sup>-1</sup> ]
$Q$	discharge	$= \text{func}(h_S)$	[mm h <sup>-1</sup> ]
<b>Internal fluxes</b>			
$P_S$	precipitation into surface water reservoir	$= P \cdot a_S$	[mm h <sup>-1</sup> ]
$P_V$	precipitation into vadose zone	$= P \cdot (1 - W) \cdot a_G$	[mm h <sup>-1</sup> ]
$P_Q$	precipitation into quickflow reservoir	$= P \cdot W \cdot a_G$	[mm h <sup>-1</sup> ]
$ET_V$	actual evapotranspiration vadose zone	$= ET_{\text{pot}} \cdot \beta \cdot a_G$	[mm h <sup>-1</sup> ]
$ET_S$	actual evapotranspiration surface water	$= ET_{\text{pot}} \cdot a_S$	[mm h <sup>-1</sup> ]
$f_{GS}$	groundwater drainage/surface water infiltration	$= \frac{(c_D - d_G - h_S) \cdot \max((c_D - d_G), h_S)}{c_G} \cdot a_G$	[mm h <sup>-1</sup> ]
$f_{QS}$	quickflow	$= \frac{h_Q}{c_Q} \cdot a_G$	[mm h <sup>-1</sup> ]
<b>Model parameters</b>			
$c_W$	wetness index parameter		[mm]
$c_V$	vadose zone relaxation time		[h]
$c_G$	groundwater reservoir constant		[mm h]
$c_Q$	quickflow reservoir constant		[h]
<b>Supplied parameters</b>			
$a_S$	surface water area fraction		[-]
$a_G$	groundwater reservoir area fraction	$= 1 - a_S$	[-]
$c_D$	channel depth		[mm]
<b>User-defined functions with defaults</b>			
$W(d_V)$	wetness index	$= \cos\left(\frac{\max(\min(d_V, c_W), 0) \cdot \pi}{c_W}\right) \cdot \frac{1}{2} + \frac{1}{2}$	[-]
$\beta(d_V)$	evapotranspiration reduction factor	$= \frac{1 - \exp[\zeta_1(d_V - \zeta_2)]}{1 + \exp[\zeta_1(d_V - \zeta_2)]} \cdot \frac{1}{2} + \frac{1}{2}$	[-]
$d_{V,eq}(d_G)$	equilibrium storage deficit	$= \theta_s \left( d_G - \frac{d_G^{1-1/b}}{(1-1/b)\psi_{ae}^{-1/b}} - \frac{\psi_{ae}}{1-b} \right)$	[mm]
$Q(h_S)$	stage–discharge relation	$= c_S \left( \frac{h_S - h_{S,\min}}{c_D - h_{S,\min}} \right)^{x_S}$	[mm h <sup>-1</sup> ]
<b>Parameters for default functions</b>			
$\zeta_1$	curvature ET reduction function		[-]
$\zeta_2$	translation ET reduction function		[mm]
$b$	pore size distribution parameter		[-]
$\psi_{ae}$	air entry pressure		[mm]
$\theta_s$	soil moisture content at saturation		[-]
$c_S$	surface water parameter: bankfull $Q$		[mm h <sup>-1</sup> ]
$x_S$	stage–discharge relation exponent		[-]
$h_{S,\min}$	surface water level when $Q = 0$		[mm]



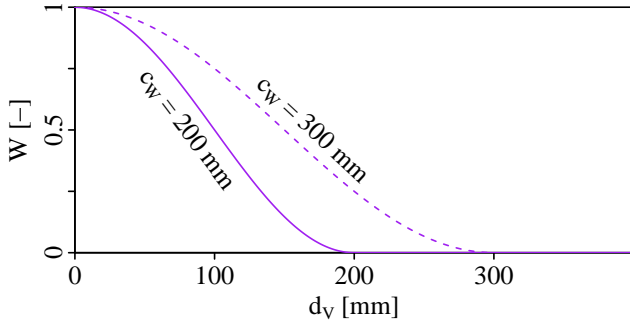


Figure 2.2: Relation between wetness index ( $W$ ) and storage deficit ( $d_V$ ) for two wetness index parameter values ( $c_W$ ). The value of the wetness index parameter is a threshold value: when the storage deficit exceeds  $c_W$ , no precipitation will be lead to the quickflow reservoir.

parameter (see Sec. 2.8). In the following sections the processes occurring within and between each compartment are discussed.

Because the soil reservoir has no lower boundary and the surface water reservoir no upper boundary, the groundwater depth  $d_G$  is measured with respect to the soil surface and the surface water level  $h_S$  with respect to the channel bottom. The channel bottom  $c_D$ , with respect to the soil surface, is used to compute the difference in level, which is necessary for the computation of groundwater drainage. The quickflow reservoir level  $h_Q$  is measured with respect to the bottom of that reservoir. The storage deficit  $d_V$  is an effective thickness, instead of a level or depth.

## 2.2 Precipitation and wetness index

Precipitation ( $P$ ) is divided between the 3 reservoirs: a fixed fraction  $a_S$  falls directly onto the surface water ( $P_S$ ) and the remainder is divided between the vadose zone ( $P_V$ ) and the quickflow reservoir ( $P_Q$ ). The wetness index ( $W$ ) gives the fraction of the rainfall that is led to the quickflow reservoir and ranges from 0 (dry – all water is led to the soil reservoir) to 1 (wet – all water is led to the quickflow reservoir). The wetness index is a function of storage deficit ( $d_V$ , Sec. 2.4). This relation can be supplied by the user (see Sec. 4.4), but as default a cosine function has been implemented (see Fig. 2.2), which starts at 1 when the soil is completely saturated ( $d_V = 0$ ) and drops to zero when  $d_V$  is equal to the wetness parameter  $c_W$  [mm], which has to be calibrated:

$$W = \cos \left( \frac{\max(\min(d_V, c_W), 0) \cdot \pi}{c_W} \right) \cdot \frac{1}{2} + \frac{1}{2}. \quad (2.1)$$

A negative value of  $d_V$  can occur in rare cases of large-scale ponding (Sec. 2.11). Note that in WALRUS, ponding and overland flow can only be caused by saturation excess and infiltration excess is not considered.

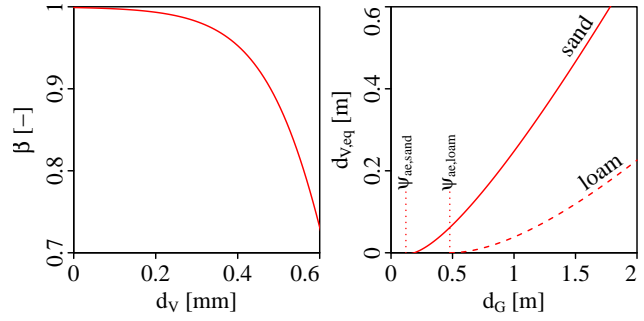


Figure 2.3: Default functions. Left: relation between evapotranspiration reduction ( $\beta$ ) and storage deficit ( $d_{V,eq}$ ). Right: relation between equilibrium storage deficit ( $d_G$ ) and groundwater depth for two soil types, based on Brooks–Corey equilibrium soil moisture profiles (Clapp and Hornberger, 1978).

## 2.3 Evapotranspiration

Evapotranspiration (ET) takes place from the surface water reservoir ( $ET_S$ ) and the vadose zone ( $ET_V$ ). The actual evapotranspiration from the vadose zone depends on the potential evapotranspiration rate and the storage deficit (Fig. 2.3). The relation between the evapotranspiration reduction factor  $\beta$  and the storage deficit can be supplied by the user (see Sec. 4.4). As a default, a two-parameter function has been implemented:

$$\beta = \frac{ET_{act}}{ET_{pot}} = \frac{1 - \exp[\zeta_1(d_V - \zeta_2)]}{1 + \exp[\zeta_1(d_V - \zeta_2)]} \cdot \frac{1}{2} + \frac{1}{2}. \quad (2.2)$$

The evapotranspiration reduction factor approaches one (no reduction) when the soil is saturated and decreases with storage deficit: first slowly, then more quickly and then more slowly again (although this end of the curve is never reached in practice). Eq. (2.2) has two parameters:  $\zeta_1$  determines the curvature and  $\zeta_2$  determines at which value of  $d_V$  the reduction factor is 0.5 (the inflection point). Note that Eq. (2.2) does not account for the effects of waterlogging on transpiration, although the net effect on ET is likely limited because of the compensating effect of soil evaporation. In addition, under extremely dry conditions Eq. (2.2) will overestimate the soil moisture stress, but such conditions approach the limits of the range for which the assumptions behind WALRUS are valid.

The open water evaporation is assumed to be equal to the potential evapotranspiration ( $ET_{pot}$ ) of a well-watered soil. A Penman approximation would be more appropriate, but for most catchments only one estimate of evapotranspiration is available. In addition, the area fraction of open water and consequently the error is small. No evapotranspiration from the surface water occurs when the surface water reservoir is empty. Because the groundwater and surface water reservoirs together cover the entire catchment area, no evapotranspiration occurs from

Table 2.2: Parameters of the Brooks–Corey equilibrium soil moisture profile. The first 11 rows are taken from Clapp and Hornberger (1978). The last two lines are obtained from combined soil moisture and groundwater observations in two experimental catchments.

Soil type	$b$ [–]	$\psi_{ae}$ [mm]	$\theta_s$ [–]
Sand	4.05	121	0.395
Loamy sand	4.38	90	0.410
Sandy loam	4.90	218	0.435
Silt loam	5.30	786	0.485
Loam	5.39	478	0.451
Sandy clay loam	7.12	299	0.420
Silt clay loam	7.75	356	0.477
Clay loam	8.52	630	0.476
Sandy clay	10.40	153	0.426
Silty clay	10.40	490	0.492
Clay	11.40	405	0.482
Hupsel	2.63	90	0.418
Cabauw	16.77	9	0.639

the quickflow reservoir. The effect of vegetation diversity on potential evapotranspiration can be accounted for by preprocessing.

## 2.4 Storage deficit

The dryness of the vadose zone is expressed by the storage deficit ( $d_V$ ), representing the volume of empty soil pores per unit area, or in other words, the depth of water necessary to reach saturation. The vertical profile of soil moisture is not simulated explicitly and, as WALRUS is a lumped model, neither is its horizontal variability. The storage deficit controls the precipitation division between groundwater and quickflow ( $W$ ), evapotranspiration reduction ( $\beta$ ) and the change in groundwater depth ( $d_G$ ) and is itself the result of all fluxes into or out of the soil reservoir, both the vadose zone and the groundwater zone.

In the field, time series of storage deficit ( $d_V$ ) can be estimated from soil moisture ( $\theta$  [–]) profile data. For each depth the soil moisture content at saturation ( $\theta_s$  [–]) has to be determined, which can often be done by taking the highest measured soil moisture content at that depth. The difference between the profiles of  $\theta$  and  $\theta_s$  gives the profile of the fraction of soil filled with air (and the remainder,  $1 - \theta_s$ , gives the soil particle fraction). The storage deficit is obtained by integrating this air profile over depth  $d$  from the groundwater table  $d_G$  to the soil surface:

$$d_V = \int_0^{d_G} (\theta_s - \theta) \, dd. \quad (2.3)$$

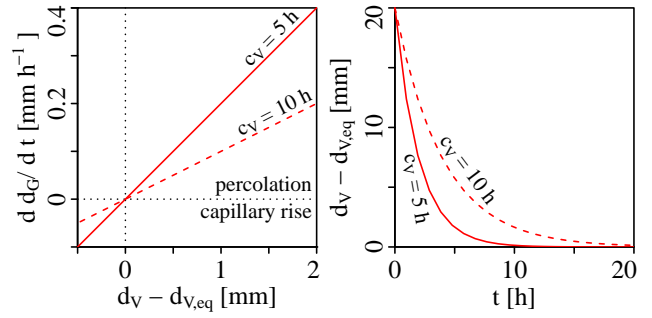


Figure 2.4: Relation between the change in groundwater depth ( $dd_G/dt$ ) and the difference between storage deficit ( $d_V$ ) and equilibrium storage deficit ( $d_{V,eq}$ ) for two vadose zone relaxation time values ( $c_V$ ). The right Figure shows how the difference between storage deficit and equilibrium storage deficit will decrease in time when all other model variables remain the same.

## 2.5 Equilibrium storage deficit

For every groundwater depth  $d_G$ , an equilibrium soil moisture profile exists where at all depths gravity is balanced by capillary forces, and no flow occurs. From this profile the equilibrium storage deficit  $d_{V,eq}$  can be derived in the same way as  $d_V$ , namely by integrating the volume of empty soil pores over depth. The relation between  $d_{V,eq}$  and  $d_G$  can be estimated from combined observations of groundwater and soil moisture. By assuming that on average  $d_{V,eq}$  equals  $d_V$ , the relation can be read from a  $(d_G, d_V)$ -plot and supplied to WALRUS (see Sec. 4.4).

Alternatively, one can assume a relation based on parametrisations of steady-state (i.e. no-flow) profiles reported by e.g. Brooks and Corey (1964) and Van Genuchten (1980). WALRUS uses the power law of Brooks and Corey as default because it requires only two parameters. The profile of soil moisture content  $\theta$  [–] as a function of height above the groundwater table  $h$  [mm] according to Clapp and Hornberger (1978) is

$$\theta = \theta_s \left( \frac{h}{\psi_{ae}} \right)^{-1/b}, \quad (2.4)$$

with  $b$  the pore size distribution parameter [–] and  $\psi_{ae}$  the air entry pressure [mm]. The air entry pressure raises the power law distribution above the groundwater table to allow for the capillary fringe (the saturated area above the groundwater table). The parameters  $b$ ,  $\psi_{ae}$  and  $\theta_s$  differ per soil type and selected results from laboratory experiments by Clapp and Hornberger (1978) are given in Table 2.2 (see Cosby et al., 1984, for interpolations between soil types). When the part of the profile between the capillary fringe and the soil surface from Eq. (2.4) is substituted in Eq. (2.3), the relation between equilibrium

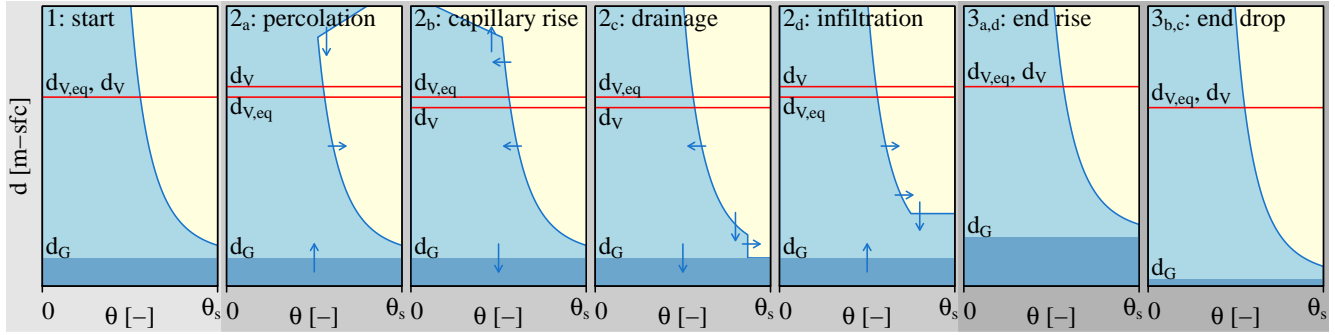


Figure 2.5: Illustration of the four scenarios for change in groundwater levels: **(a)** percolation after rainfall, **(b)** capillary rise after evapotranspiration, **(c)** percolation after drainage and **(d)** capillary rise after infiltration. WALRUS only simulates the solid lines of  $d_G$ ,  $d_V$  and  $d_{V,eq}$  rather than the profiles of relative saturation (dashed). The areas right of the curves (the integral of  $(\theta_s - \theta)$  over  $d$ ) is equal to the values of  $d_V$ .

storage deficit and groundwater depth becomes

$$\begin{aligned} d_{V,eq} &= \int_{\psi_{ae}}^{d_G} \left[ \theta_s - \theta_s \left( \frac{h}{\psi_{ae}} \right)^{-1/b} \right] dh \\ &= \theta_s \left( d_G - \frac{d_G^{1-1/b}}{(1 - \frac{1}{b})\psi_{ae}^{-1/b}} - \frac{\psi_{ae}}{1-b} \right). \end{aligned} \quad (2.5)$$

Heterogeneities, such as soil layering or disruption by plant roots, macrofauna and human activity, cause differences between laboratory and field observations.

## 2.6 Percolation and capillary rise

In practice, the soil moisture profile and storage deficit are never perfectly in equilibrium with the groundwater depth. Addition (e.g. through precipitation) and removal (e.g. by drainage or evapotranspiration) of water cause an imbalance between gravity and capillary forces, leading to downward (percolation) or upward (capillary rise) flow towards a new equilibrium situation. Because the flow decreases with proximity to the equilibrium, this equilibrium will only be reached asymptotically.

The exact profile of relative saturation is not simulated explicitly in WALRUS, but the temporal dynamics of  $d_V$  and  $d_G$  caused by the interactions between groundwater and vadose zone are taken into account. The groundwater depth responds to changes in storage deficit. The change in groundwater depth is parameterised as a function of the difference between the actual storage deficit (computed from the water budget in the soil reservoir) and the equilibrium storage deficit corresponding to the current groundwater level:

$$\frac{dd_G}{dt} = \frac{d_V - d_{V,eq}}{c_V}, \quad (2.6)$$

with  $c_V$  the vadose zone relaxation time constant, which determines how quickly the system advances towards a new equilibrium (Fig. 2.4).

Four situations may occur (illustrated in Fig. 2.5).

- (1) Water is added to the vadose zone through percolation. The actual storage deficit is smaller than the equilibrium for the current groundwater depth. Water will flow downward and the groundwater level will rise gradually to the depth corresponding to the actual storage deficit.
- (2) Water is removed from the vadose zone through evapotranspiration. The actual storage deficit exceeds the equilibrium for the current groundwater depth. Water will flow upward to replenish the shortage in the top soil and the groundwater level will drop gradually.
- (3) Water is removed from the soil reservoir through drainage, downward seepage or groundwater extraction. Air is sucked into the soil and the actual storage deficit increases. This happens instantaneously, because water is incompressible. Water will percolate to reach an equilibrium profile again and the groundwater level will drop gradually.
- (4) Water is added to the soil reservoir through infiltration from surface water or upward seepage. The storage deficit decreases directly and the groundwater table rises gradually.

## 2.7 Groundwater

Drainage depends on the difference in water level between the surface water and groundwater reservoirs (rather than on groundwater levels alone), allowing for feedbacks and infiltration of surface water into the soil. Drainage of groundwater towards the surface water reservoir or infiltration of surface water  $f_{GS}$  is computed as

$$f_{GS} = \frac{(c_D - d_G - h_S) \cdot \max((c_D - d_G), h_S)}{c_G} \cdot a_G, \quad (2.7)$$

with  $d_G$  the depth of the groundwater table below the soil surface,  $c_G$  a reservoir constant [mm h] and  $c_D$  the average channel depth [mm] (see also Table 2.1 and Fig. 2.1). The parameter  $c_G$  represents the combined effect of all resistance and variability therein and depends on soil type (hydraulic conductivity) and drainage density. For the effect of  $c_G$  and  $h_S$  on groundwater drainage, see Fig. 2.6.

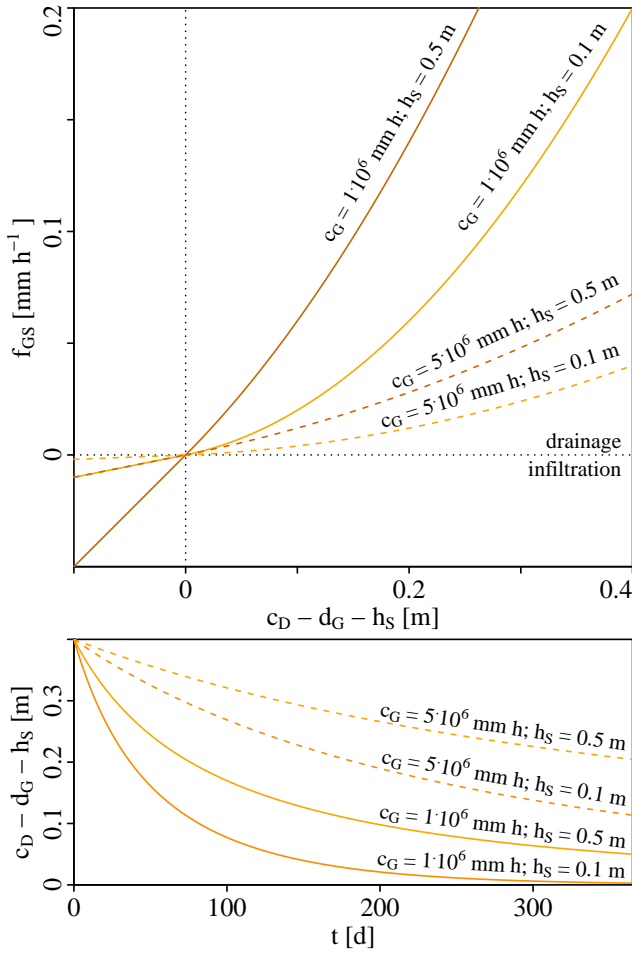


Figure 2.6: Relation between the drainage or infiltration flux ( $f_{GS}$ ) and the difference between groundwater level above the channel bottom ( $c_D - d_G$ ) and surface water level ( $h_S$ ) for two groundwater reservoir constant values ( $c_G$ ) and two surface water level values ( $h_S$ ). The bottom Figure shows how the difference between groundwater level and surface water level will decrease in time when all other model variables remain the same.

The first term of Eq. (2.7),  $c_D - d_G - h_S$ , expresses the pressure difference driving the flow. The second term,  $\max((c_D - d_G), h_S)$ , expresses the contact surface (parameterised as a depth) through which the flow takes place. These terms can be compared to the pressure head difference and layer thickness commonly used in groundwater models. The contact surface-term accounts for decreasing drainage efficiency when groundwater and surface water levels drop and headwaters run dry. With this term, the variable source area concept (Beven and Kirkby, 1979) is implemented effectively and without additional parameters.

When groundwater drops below the surface water level, infiltration will be computed with the same relation, decreasing to zero when the surface water reservoir is empty (the second term  $\max((c_D - d_G), h_S)$  becomes zero). The same parameter  $c_G$  is used for both ground-

water drainage and surface water infiltration to limit the number of parameters, even though the resistance may be different in practice.

## 2.8 Quickflow

The quickflow reservoir simulates the combined effect of all water flowing through quick flow paths towards the surface water: overland, macropore and drainpipe flow. This reservoir can therefore be seen as a collection of ponds, small drainage trenches or gulleys, soil cracks, animal burrows and drainpipes. Quickflow  $f_{QS}$  depends linearly on the elevation of the water level in the quickflow reservoir  $h_Q$ , with a time constant (reservoir constant)  $c_Q$  (Fig. 2.7):

$$f_{QS} = \frac{h_Q}{c_Q} \cdot a_G. \quad (2.8)$$

Water cannot flow from the surface water into the quickflow reservoir. Therefore, a sudden surface water level rise caused by an increase in surface water supply or weir elevation does not affect the quickflow reservoir directly.

The water level in the quickflow reservoir cannot be coupled to measurable variables directly – groundwater level measurements show the combined effect of the seasonal variation of the groundwater depth and the high resolution dynamics of the quickflow reservoir. Even though quickflow is parameterised as a single linear reservoir, it is essential to include this reservoir to mimic the large and variable contribution of these flowroutes.

## 2.9 Surface water

In WALRUS, surface water forms an integral part of the model structure. The surface water level  $h_S$  represents the water level in the average channel with respect to the channel bottom. The distance between channel bottom and soil surface  $c_D$  is calibrated or estimated from field observations. The stage-discharge relation  $Q = \text{func}(h_S)$

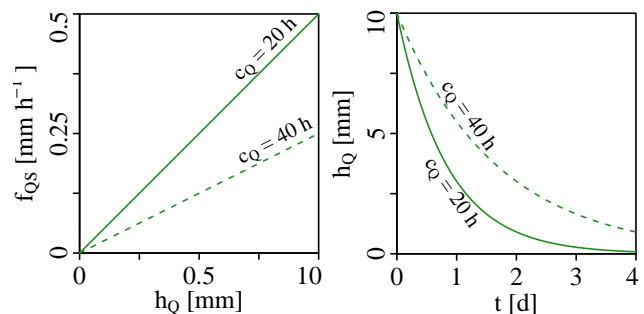


Figure 2.7: Relation between quickflow ( $f_{QS}$ ) and quickflow reservoir level ( $h_Q$ ) for two quickflow reservoir constant values ( $c_Q$ ). The right Figure shows how the quickflow reservoir level will decrease in time when all other model variables remain the same.

specifies the relation between surface water level and discharge at the catchment outlet (in  $\text{mm h}^{-1}$ ). It is provided by the user as a function (see Sec. 4.4), e.g. the relation belonging to the weir at the catchment outlet, or as a lookup table<sup>1</sup>. A threshold level  $h_{S,\min}$  can be included in the stage–discharge relation to account for a weir or other water management structures. If applicable, a value or time series of  $h_{S,\min}$  should be provided. When the surface water level drops below the crest of a weir, discharge will be zero, but because there may still be drainage, infiltration and evaporation, it is important to include standing water. A default stage–discharge relation with the shape of a power law with a default exponent  $x_S$  of 1.5 has been implemented:

$$Q = c_S \left( \frac{h_S - h_{S,\min}}{c_D - h_{S,\min}} \right)^{x_S} \quad (2.9)$$

for  $h_S \leq c_D$ . The default exponent value 1.5 for  $x_S$  is inspired by equilibrium flow in open channels (Manning, 1889). The parameter  $c_S$  corresponds to the discharge at the catchment outlet (in  $\text{mm h}^{-1}$ ) when the surface water level reaches the soil surface, comparable to the bankfull discharge. It can be calibrated or provided based on field observations.

## 2.10 Seepage and surface water supply

All fluxes across the catchment boundary, except for the discharge at the catchment outlet, are combined in the external groundwater flow term  $f_{XG}$  (downward or upward seepage, lateral groundwater inflow or outflow or groundwater abstraction) and the external surface water flow term  $f_{XS}$  (supply or extraction). Positive values denote flow into the catchment. If applicable, time series of  $f_{XG}$  or  $f_{XS}$  should be provided by the user. Seepage and surface water supply are not parameterized, because they do not depend on processes within the catchment. In case no data are available, a groundwater or surface water management model could be used to obtain time series of seepage and surface water supply, which can be used as input for WALRUS.

Because these fluxes are added to the soil reservoir or surface water reservoir, they influence other variables through the different feedbacks implemented in the model. Most parametric rainfall–runoff models do not contain a surface water reservoir and therefore surface water supply can only be added to discharge afterwards and the impact of surface water increase on groundwater level and the groundwater drainage flux is not considered.

## 2.11 Large-scale ponding and flooding

The quickflow reservoir simulates the effect of local ponding and overland flow, but large-scale ponding may also occur. When the storage deficit becomes zero (i.e. all soil pores are filled with water), the groundwater level will rise directly to the surface (as observed by Gillham, 1984; Brauer et al., 2011). Storage deficit and groundwater depth continue to drop (i.e. become more negative) together as there are no capillary forces any more and water level and pressure head coincide – negative  $d_V$  and  $d_G$  express ponding depths. Note that the levels rise less quickly above ground as the storativity becomes 1.

Unfortunately, few quantitative, catchment-scale observations exist of different fluxes during floods. Because WALRUS has no spatial dimensions, the complex process of overland flow must be simplified. It is assumed that when the groundwater or surface water level rises above the soil surface, the groundwater drainage/surface water infiltration flux  $f_{GS}$  will include overland flow and is instantaneous, because overland flow is much faster than groundwater flow. When the surface water level exceeds the soil surface, discharge becomes less sensitive to changes in surface water level, represented by an abrupt change in the stage–discharge relation. However, as soon as the surface water level exceeds the soil surface, the excess water is led to the soil reservoir directly and therefore  $h_S$  hardly rises above the soil surface. Therefore, we keep the same stage–discharge relation when  $h_S > c_D$  as a default. When the modelled groundwater table reaches the soil surface, an abrupt change in catchment discharge occurs. This is in contrast to the gradual activation of different flowpaths when the catchment effective groundwater table is below surface (as represented by the wetness index).

We investigated the option of making the surface water area fraction  $a_S$  a function of  $h_S$ , representing gradual widening of brooks and inundation of areas close to the surface water network, and thereby smoothing the effect of flooding on discharge at the catchment outlet. Unfortunately, this approach made the model structure less intuitive and introduced more degrees of freedom to define the shape of this function. Because flooding of the surface water reservoir only occurs during extremely wet situations, we chose to keep the model structure simple and leave  $a_S$  fixed.

## 2.12 Outlook: possible model extensions

Some processes are not taken into account in the core model yet, but a user could easily add preprocessing and postprocessing steps to adapt WALRUS to catchment-specific situations. (1) The potential evapotranspiration estimated at a meteorological station may not be representative for the collection of vegetation types in the

<sup>1</sup>The lookup table is not implemented in the current version of WALRUS. If necessary, a user can transform the lookup table to a function using the `approxfun` function in R.

catchment. Therefore, one could use land cover distributions and crop factors to determine the catchment average potential evapotranspiration. (2) Currently, WALRUS is set-up to receive liquid precipitation, but preprocessing steps to account for snow and/or interception can be added. For example, the delay in precipitation input caused by snow accumulation and melt can be simulated with methods based on the land surface energy balance (Kustas et al., 1994) or a degree-day method (Seibert, 1997). (3) Interception can be parameterised with a threshold. Only the rainfall which exceeds the threshold is used as input for the model. The intercepted water evaporates directly and is not subtracted from  $ET_{pot}$  (Teuling and Troch, 2005). (4) Paved surfaces have a low infiltration capacity, which limits groundwater recharge. This can be parametrized by decreasing the groundwater reservoir area  $a_G$ , introducing a paved surface area and leading the fraction of the rainfall belonging to this area directly to the surface water. (5) For large catchments, the discharge pulse from the model can be delayed and attenuated in the channels. It is possible to add a routing function to account for the delay and attenuation.

Another possibility is to couple WALRUS to other models. This is ongoing research and not implemented in the current version of WALRUS. The outflow  $Q$  of one catchment can be used as surface water supply  $f_{XS}$  for another WALRUS-unit downstream. With this technique, one could make a chain of WALRUS units (effectively, building a spatially distributed model) to simulate sub-catchments (with possibly different catchment characteristics and therefore parameter values) separately. Groundwater flow from one unit to the next can be computed from groundwater levels in adjacent cells and Eq. (2.7). This groundwater flow is added to or subtracted from the seepage flux  $f_{XG}$  for both units. Regional groundwater flow from a distributed groundwater model can be added to or subtracted from the soil reservoir through the seepage flux  $f_{XG}$ . This can be specified with a time series or an external groundwater level. The outflow of the model can be used as input for a hydraulic model. Discharge from an upstream catchment as computed from a hydraulic model can also be used as input  $f_{XS}$ .



## 3 | Model implementation

In this Chapter we describe some key parts of the model implementation, which affect the model application and performance. This Chapter is largely based on Section 6 of the paper in GMD (Brauer et al., 2014a). You can decide to skip this Chapter if you want to get WALRUS running first and learn about what happens in the background later.

### 3.1 Code set-up

The model code is written in R and consists of several functions. Two functions form the core of the model code: `WALRUS_loop` and `WALRUS_step`. In `WALRUS_loop` the initial conditions are set, a for-loop over each time step is run and output data are organized. For every time step, the function `WALRUS_step` is called, which contains the actual model computations. Some additional functions provide help by preprocessing forcing data, setting default parameters, and postprocessing of the model output: figures, water balance computations and analysis of residuals. Some example scripts provide templates in which functions are called for preprocessing, calibrating, running the model and postprocessing.

### 3.2 Initial conditions

The model can (as default) compute initial conditions for all states automatically, based on a stationary situation (thereby avoiding long burn-in periods). The quickflow reservoir is initially empty. The initial surface water level is derived from the first discharge observation and the stage–discharge relation. The initial groundwater depth is computed with the assumption that initial groundwater drainage ( $f_{GS}$ ) is equal to the initial discharge (see Sec. for 6.3 examples). The initial storage deficit  $d_V$  is the equilibrium value belonging to the initial groundwater depth  $d_G$ .

A user can also supply initial values for the four states:  $d_{V,0}$ ,  $d_{G,0}$ ,  $h_{Q,0}$ ,  $h_{S,0}$ . If some states are supplied, but not all, the remaining states will be computed from the supplied ones and equilibrium conditions. For example, if only the initial groundwater depth is supplied (or calibrated) by the user,  $h_{Q,0}$  is computed such that  $Q_0 = f_{GS,0} + f_{QS,0}$  again.

It is also possible to supply the fraction of the initial discharge originating from drainage  $G_{frac}$  and the model will solve

$$Q_0 \cdot G_{frac} = \frac{(c_D - d_{G,0} - h_{S,0}) \cdot (c_D - d_{G,0})}{c_G} \quad (3.1)$$

for  $d_{G,0}$  with the quadratic formula and then use the remainder of the discharge to compute the initial quickflow

reservoir level:

$$h_{Q,0} = Q_0 \cdot (1 - G_{frac}) \cdot c_Q. \quad (3.2)$$

Finally, the initial discharge  $Q_0$  can be supplied, which will be used to compute  $h_S$ .

A user can choose to use a warming-up period in case the uncertainty around the initial conditions is large. It is implemented in the code, but by default, the warming-up period is set to zero.

### 3.3 Parameters

WALRUS has four parameters which require calibration:  $c_W$ ,  $c_V$ ,  $c_G$  and  $c_Q$ . We found that  $c_W$ ,  $c_G$  and  $c_Q$  are well identifiable in the discharge time series, but that  $c_V$  is more difficult to estimate (Brauer et al., 2014b). Parameter dependence is fortunately limited (Brauer et al., 2014b).

The parameters have a physical meaning and can be explained qualitatively with catchment characteristics. The channel depth  $c_D$  and surface water area fraction  $a_S$  can be estimated from field observations. When the default stage–discharge relation is used, the bankfull discharge  $c_S$  and (if applicable) the weir elevation  $h_{S,min}$  need to be supplied (or calibrated) as well.

Parameters are catchment-specific, but time-independent, to allow a calibrated model to be run for both long periods and events. We did not implement a specific calibration routine in the model, but used the HydroPSO package, which is a particle swarm optimization technique (Zambrano-Bigarini and Rojas, 2013). Another option is to use the Levenberg-Marquardt algorithm as implemented in the `minpack`-package. The user can define the (multi-)objective function.

### 3.4 Forcing

Forcing data can be supplied as a time series or as a function (e.g. a sine function for  $ET_{pot}$  or a Poisson rainfall generator). Observation times do not need to be equidistant, which is especially useful for tipping-bucket rain gauges. Forcing time series are converted to functions (e.g. cumulative  $P$  as function of time), which allows other time steps than used for the original forcing.

### 3.5 If-statements

If-statements associated with thresholds can cause nonlinearities in models and their abrupt changes hamper calibration, in particular when using gradient-based methods. It is therefore important to know that there are four

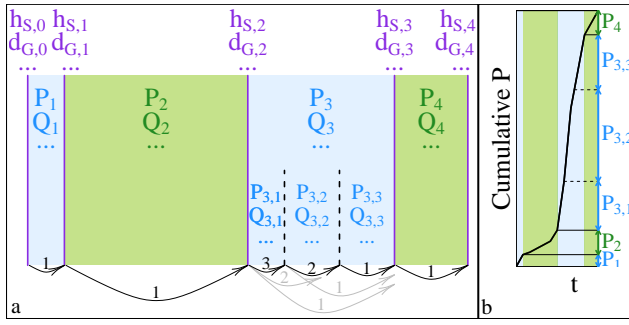


Figure 3.1: Illustration of the variable time step procedure. **(a)** The non-equidistant output time steps (purple) are used as first attempts for computations of fluxes (blue/green) and states (purple), but during time step number 3, the precipitation sum is too large (panel **b**) and the step is divided into substeps: it is halved and then halved again until the criterion was reached. Note that even though the size of output time step 2 is larger, it is not divided into substeps, because all criteria are met.

causes for abrupt changes in the model: (1) the stage–discharge relation (supplied by a user) may show abrupt changes at the elevation of the crest of the weir or at the soil surface; (2) no evaporation occurs from empty channel beds; (3) if the storage deficit becomes negative or exceeds the groundwater depth, the groundwater depth becomes equal to the storage deficit; (4) if either groundwater or surface water level exceeds the soil surface, overland flow is instantaneous.

### 3.6 Integration scheme

The model is implemented as an explicit scheme, because nonlinearities caused by feedbacks and if-statements do not allow for the use of an implicit scheme. An advantage of the use of an explicit scheme is that the mass balance will always close. The states at the end of the previous time step are used to compute the fluxes during the current time step, which are then used to compute the states at the end of the current time step (Fig. 3.1). The output data file lists the sums of the fluxes during and the states at the end of each time step.

### 3.7 Time step

The user can specify at which moments output should be generated, for example with a fixed interval (i.e. each hour or day; implemented in the current version of the WALRUS package), with increased frequency during certain events or after a fixed amount of rainfall (not implemented in the current version of the WALRUS package yet). The output time steps can be both larger and smaller than those of the forcing.

An important feature of the model code is the flexible computation time step. The model first attempts to

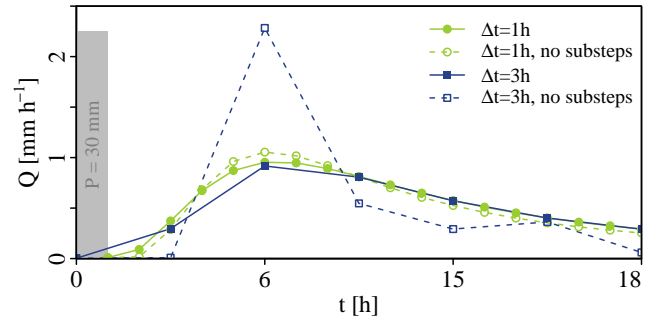


Figure 3.2: The effect of variable time steps on the model output. An artificial case with a rainfall event of 30 mm in the first hour and no evapotranspiration. The lines connect the discharge modelled at the end of a time step (instantaneous value), and do not represent the sum over the time step (which is given in the output file). used.

run a whole output time step at once, but the time step is decreased when (1) the rainfall sum, discharge sum or change in discharge, surface water level or groundwater depth during the time step exceeds a certain threshold, or when (2) the surface water level is negative at the end of the time step. The first criterion prevents numerical instability caused by the explicit integration scheme and a delayed the response to rainfall as a result of the explicit model code (it takes one step to update the surface water level and another for the discharge). The second criterion is necessary because the total surface water outflow, computed from water levels at the start of the time step and the time step size, can exceed the available water. Because this means that non-existing water flowed out, there is a physical reason to avoid this.

The procedure of decreasing time steps is illustrated in Fig. 3.1 (3rd step). First the original time step is halved and the model is run for this substep (of course with the forcing corresponding to this substep). When the criteria are still not met, the step size will be halved again and again until the criteria are met. When one substep is completed, the fluxes are stored and the states at the end of the time step are used as initial values for the next substep. Then the model is run for the remainder of the original time step and, if necessary, the substep is halved until the criteria are met. This will continue until the end of the intended output time step is reached. The sum of the fluxes of the substeps and the states of the last substep are stored in the output file (note that the mass balances are preserved).

The effect of the variable time step is illustrated in Fig. 3.2, in which the output of the model ran with a fixed time step and with variable time steps is shown. Note the erroneous time delay and magnitude of the discharge peak when no substeps are used, in particular for the three hourly time step.



### 3.8 Water balance

WALRUS is a mass conserving model, and therefore the model water budget, computed as

$$\begin{aligned} \Sigma P - \Sigma ET_{\text{act}} - \Sigma Q + \Sigma f_{\text{XG}} + \Sigma f_{\text{XS}} = \\ -\Delta d_V \cdot a_G + \Delta h_Q \cdot a_G + \Delta h_S \cdot a_S, \end{aligned} \quad (3.3)$$

always closes, although rounding errors may cause small deviations. The minus sign before  $\Delta d_V$  appears because  $d_V$  expresses a deficit and a decrease in storage deficit implies an increase in water in the reservoir. The groundwater level does not appear explicitly in the water balance, because it only plays a role as a pressure level driving groundwater drainage and surface water infiltration fluxes, while the storage deficit accounts for volume changes in the whole soil reservoir. The water balance over the whole calibration period is computed automatically (Sec. 4.5).



## 4 | The WALRUS R-package

### 4.1 R and RStudio

WALRUS is written in R. We recommend using R with a user interface called RStudio. When you are new to R and/or RStudio, you can for example read our document “A (very) short introduction to R”, which teaches you the basics of R in about an hour (Torfs and Brauer, 2015).

All WALRUS code is organised in an R package called WALRUS (see Fig. 4.1). The latest version of this package (and the user manual you’re reading) can be downloaded (see Sec. 5.2) from the first author’s GitHub: [GitHub.com/ClaudiaBrauer/WALRUS](https://github.com/ClaudiaBrauer/WALRUS) (open access). We intend to upload this package to the main R CRAN server in 2015, after the inevitable first problems have been solved. Please let us know when you encounter problems with the package, so we can improve it.

Section 5.2 contains a tutorial on how to install the package. The WALRUS package consists of several functions. Figure 4.1 shows all functions in the package. This list can be viewed in RStudio (see Sec. 5.3).

### 4.2 Core functions

The core of WALRUS (the model equations used for running the model) is split into two functions.

- `WALRUS_loop` sets initial conditions and runs a loop over all time steps. If you want to run WALRUS, you call `WALRUS_loop`. If you want to run a long time series (more than 10 years of hourly data), it is better to use `WALRUS_loop_long`, which splits the whole run into parts and uses less memory.
- `WALRUS_step` performs the calculations for each time step. This function is called by `WALRUS_loop`. The user does not need to call this function manually (this is done automatically by `WALRUS_loop`).

### 4.3 Preprocessing

There are several functions for preprocessing. These functions are to help the user set up WALRUS for a specific case. None of these functions are strictly necessary, but especially the `WALRUS_preprocessing` can be useful.

- `WALRUS_selectdates` cuts a specific period from a larger data frame which contains the forcing time series.
- `WALRUS_preprocessing` converts the forcing time series to functions. These functions yield the cumulative rainfall, potential evapotranspiration or discharge as a function of the number of seconds since 1 January 1970. This conversion is necessary for the flexible time step approach (see Sec. 3.7). The `WALRUS_preprocessing`-function will also create a

vector called `output_dates`. The dates in this vector are the moments for which output will be computed and, eventually, stored in an output file (also in seconds since 1970). In the current version of WALRUS, the preprocessing function only makes an `output_dates`-time series with fixed time steps. You can also adjust this time series or create one yourself.

- `WALRUS_snow` is a preprocessor which computes snow accumulation and melt. The output of this function is the sum of liquid precipitation and snowmelt, which is used as input for WALRUS rather than precipitation directly. This function is ran before the general preprocessor. Two methods are implemented: the degree-hour-factor method (DHF) and the shortwave radiation factor method (SRF). Additional data are necessary for the snow-preprocessor. The forcing data frame should contain a column with temperature data (called `T` and, if the method SRF is used, also a column with shortwave radiation data (column `GloRad`). Fixed values for snow parameters are implemented, which are representative for Dutch conditions (for more information on the snow parameters, see Wendt, 2015).
- `WALRUS_preprocessing_calibration` finds the observed discharge values belonging to each desired output time step (from the vector `output_dates`). These are necessary to compare to the model output during calibration. Of course, this function is only necessary when you want to calibrate.
- `WALRUS_date_conversion` converts dates in `yyyymmdd`, `yyyymmddhh` and `yyyymmddhhmm`-format to seconds since 1970. This function is run in the background, so you don’t need to call it (but you could use it, for example if you want to adjust `output_dates`).

### 4.4 Default functions

WALRUS contains four functions for which defaults have been implemented: the wetness index  $W(d_V)$  (Sec. 2.2), evapotranspiration reduction  $\beta(d_V)$  (Sec. 2.3), equilibrium storage deficit  $d_{V,eq}(d_G)$  (Sec. 2.5) and discharge  $Q(h_S)$  (Sec. 2.9). A user can replace these defaults with

- `set_func_W_dV`,
- `set_func_beta_dV`,
- `set_func_dVeq_dG` and
- `set_func_Q_hS`.

The current versions of these functions can be viewed by calling:

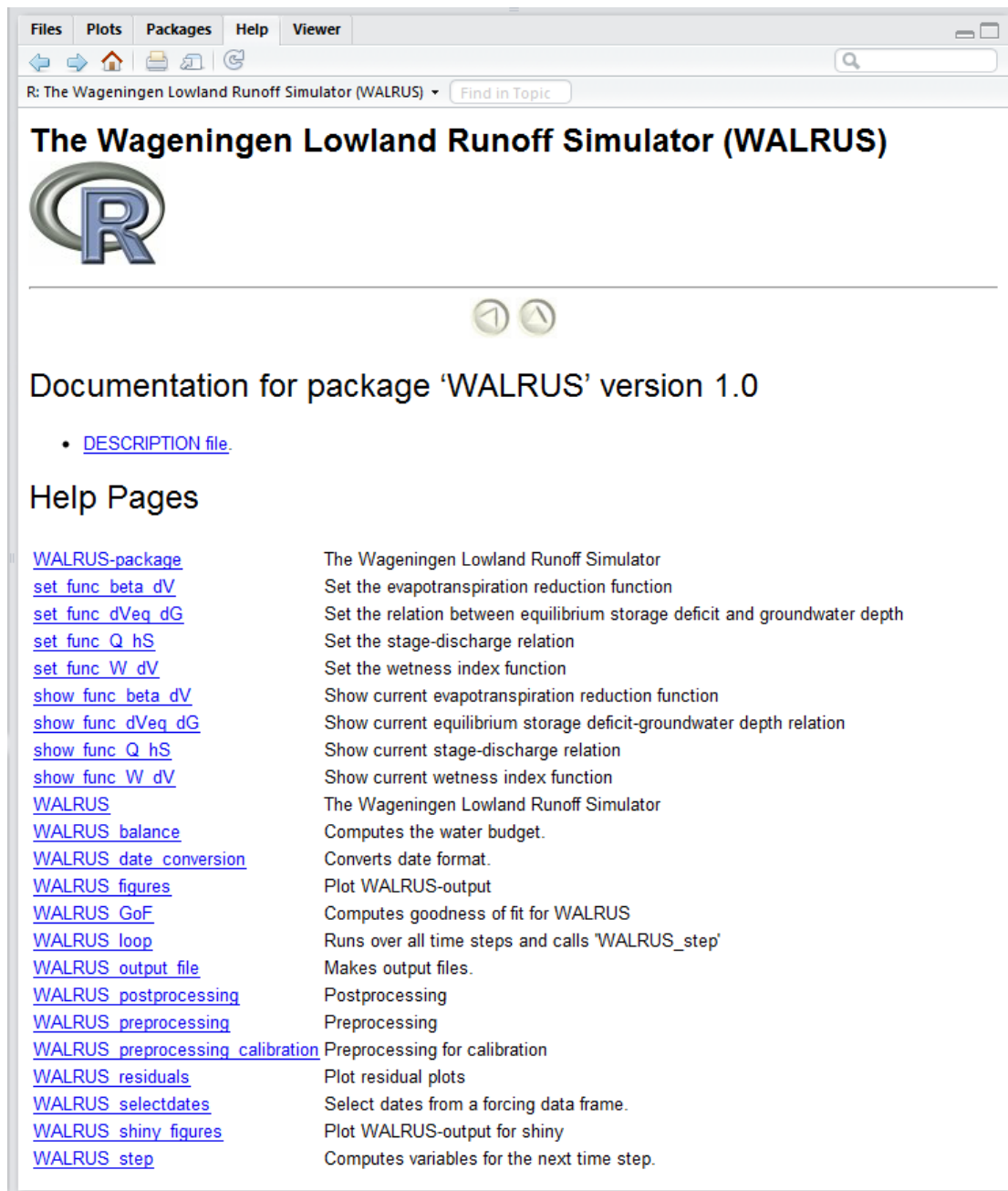


Figure 4.1: Screenshot of the WALRUS package with the functions in RStudio.

- `show_func_W_dV`,
- `show_func_beta_dV`,
- `show_func_dVeq_dG` and
- `show_func_Q_hS`.

Sec. 6.2 contains an example of how to change one of these functions.

## 4.5 Postprocessing

WALRUS does some postprocessing automatically. The function `WALRUS_postprocessing` calls 5 other functions:

- `WALRUS_output_file` makes a data file called `output_x.dat` with the time series of all (internal and external ) model variables.
- `WALRUS_GoF` – computes measures of goodness of fit: Nash-Sutcliffe efficiency of the discharge, Nash-Sutcliffe efficiency of the logarithm of the discharge (to focus on low flows) and the mean sum of squares of the discharge. These are written in the file `pars_NS_x.dat`, together with the model parameters.
- `WALRUS_balance` computes the water budget: all terms in Equation 3.3. The results are shown in

the Command Window of RStudio and stored in `balance_x.dat`.

- `WALRUS_figures` plots time series of several model variables, both shown in the Plots Window of RStudio and stored in `figures_x.pdf`. When you don't want figures, add the argument `figures=FALSE` to the `WALRUS_postprocessing`-function.
- `WALRUS_residuals` makes figures of the residuals in `analysis_residuals_x.pdf`. Residuals are only made when you add the argument `residuals=TRUE` to the `WALRUS_postprocessing`-function.

In the names above, `x` is replaced with the name a user gives to the run. In Sec. 6.1, these output files and figures are shown.

## 4.6 Graphical user interface

At the moment, we are developing a simple graphical user interface for WALRUS. This interface is built in a web application framework called Shiny ([shiny.rstudio.com](https://shiny.rstudio.com)) and will be made available on GitHub. To run Shiny, we already included a function in the package to make figures for Shiny (`WALRUS_shiny_figures`).



## 5 | Building your own WALRUS

This Chapter contains instructions and screenshots which show you how to get WALRUS running.

### 5.1 Suitability

First you should verify that WALRUS is the appropriate model choice for your catchment and your research question. In catchments with much relief, deep groundwater and little surface water, the feedbacks between unsaturated zone, saturated zone and surface water, as implemented in WALRUS, may not occur and other rainfall-runoff models may be more appropriate.

Because WALRUS has parameters which cannot be measured in the field, but require calibration, it is necessary to have some observations of either discharge or groundwater.

### 5.2 Installing the package

To run WALRUS, download and install R ([www.r-project.org](http://www.r-project.org)) and RStudio ([www.rstudio.com](http://www.rstudio.com)). Go to [www.github.com/ClaudiaBrauer/WALRUS](https://www.github.com/ClaudiaBrauer/WALRUS) and download `WALRUS_1.02.tar.gz`. In this file, all files for the WALRUS package are wrapped together.

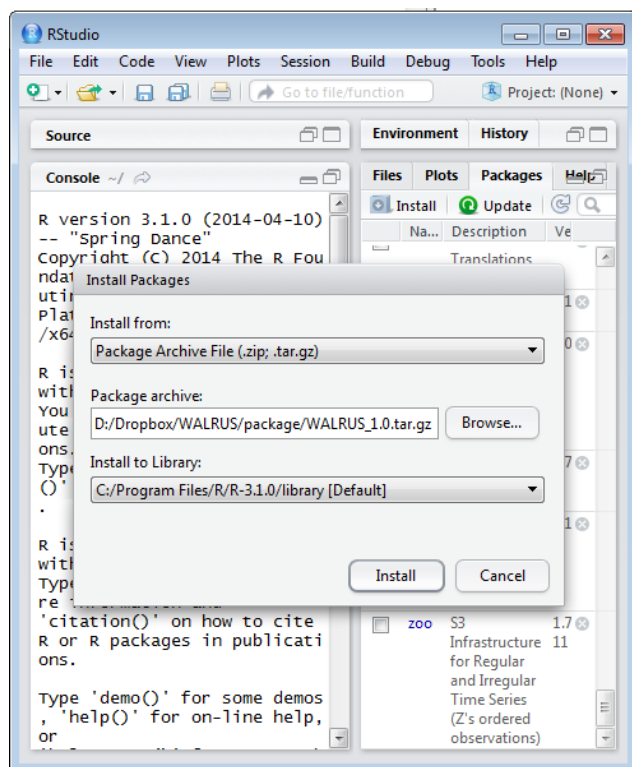


Figure 5.1: Installing the WALRUS package from a tar.gz-file downloaded from github.

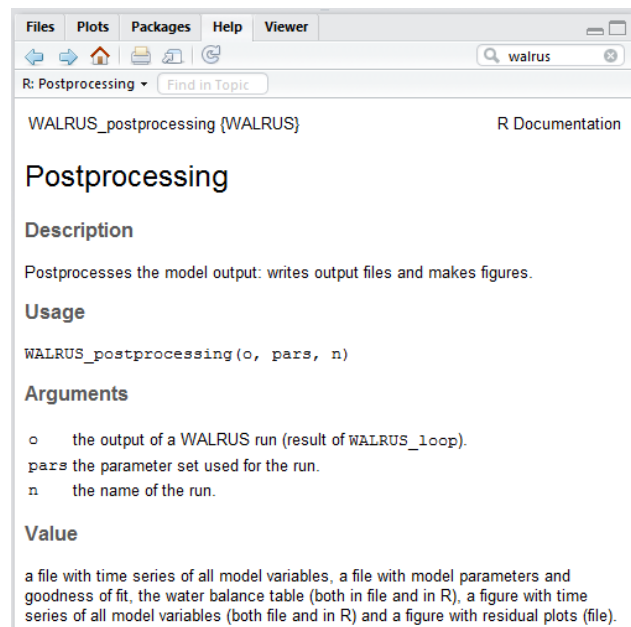


Figure 5.2: The help for one of the WALRUS functions (in the Help Window in RStudio).

To install the package in RStudio, go to the Packages Window (bottom right), click Install, choose "Install from package archive file" and browse to the file you just downloaded (see Fig. 5.1). For additional help, see our (very) short introduction to R (Torfs and Brauer, 2015). The WALRUS package uses another package called `zoo`. Install this package (click Install → click Repository (CRAN, CRANextra) → type `zoo` → click Install) before installing WALRUS.

Alternatively, the package can be installed from GitHub directly (without downloading first), using the package called `devtools`. This package is available from the R CRAN server, so you can type:

```
install.packages("devtools")
library(devtools)
install_github("ClaudiaBrauer/WALRUS")
```

Do note that (new versions of) the manual, additional documentation and other WALRUS-related information will be uploaded to GitHub as well, so you may miss these when you are downloading the package automatically.

When WALRUS is installed correctly, you will see the package appear in the list of packages in the Package Window. Checking the box in front of the package name will activate the package. You can also type `library(WALRUS)` to activate the package (it's easiest to type this at the beginning of each R-script in which you use WALRUS so it's loaded automatically).

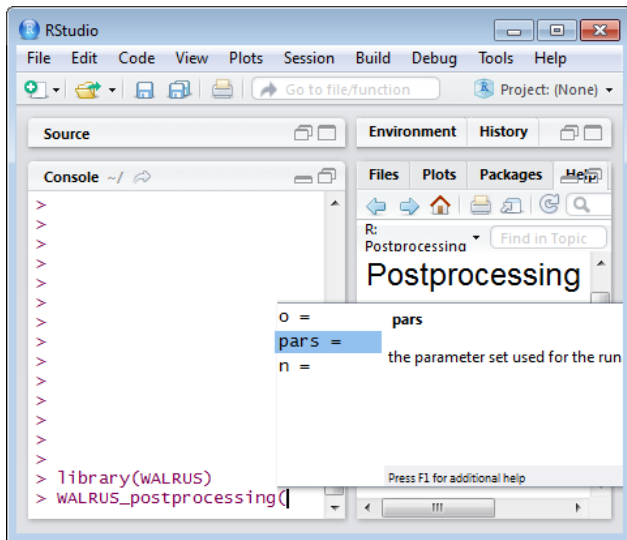


Figure 5.3: Pressing TAB will yield the options for arguments and a short description.

### 5.3 Help

If you click on the package name in the Package Window, you will get a list of functions in the WALRUS package (Fig. 4.1). Clicking on a function name will show help for that function: some basic information and required parameters (Fig 5.2). RStudio has a nice functionality: if you forgot which arguments belong to a function, you can press TAB after the opening bracket behind the function name and RStudio will show you the options (Fig 5.3). The help is still very concise – we intend to expand this much more.

### 5.4 File organization

To run WALRUS, make a folder (for each run or a few runs combined, whichever you prefer) with:

- an R-script in which you type the commands to run WALRUS (Sec. 6.1 contains an example R-script which you could use as a starting point),
- a folder with forcing data called `data`,
- an empty folder called `figures` and
- an empty folder called `output`.

It should look like Figure 5.4.

### 5.5 Data requirements

As input for WALRUS, you need:

- precipitation ( $P$ ) time series,
- potential evapotranspiration ( $ET_{\text{pot}}$ ) time series,
- discharge ( $Q$ ) or groundwater depth ( $d_G$ ) time series (for calibration),

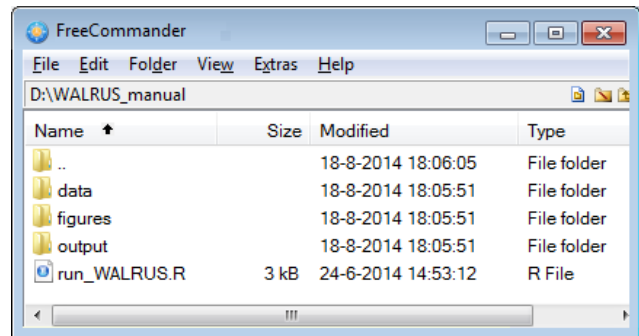


Figure 5.4: Example of the file structure with the three subfolders and the R script from which WALRUS is run.

- seepage or groundwater extraction ( $f_{\text{XG}}$ ) time series (if this is the case in your catchment) and
- surface water supply or extraction ( $f_{\text{XS}}$ ) time series (if this is the case in your catchment).

All data should be given in mm per time step. For the pre-processing function currently implemented in WALRUS, these time series should be in the same file (you can modify the code yourself of course) and the date and time should be written in the format `yyyymmdd`, `yyyymmddhh` or `yyyymmddhhmm`. Use the headers in the example in Figure 5.5. The value behind the date 2012010105 are the  $P$ ,  $ET_{\text{pot}}$  and  $Q$  sums between 5:00 and 6:00. Note that this is different from the WALRUS output file, where the values are the sums over the past period. The input data file format can be of any type R can read, e.g. `.txt`, `.csv` or `.dat`. See Section 6.1 for one example how to read the data into R and Sec. 7.2.

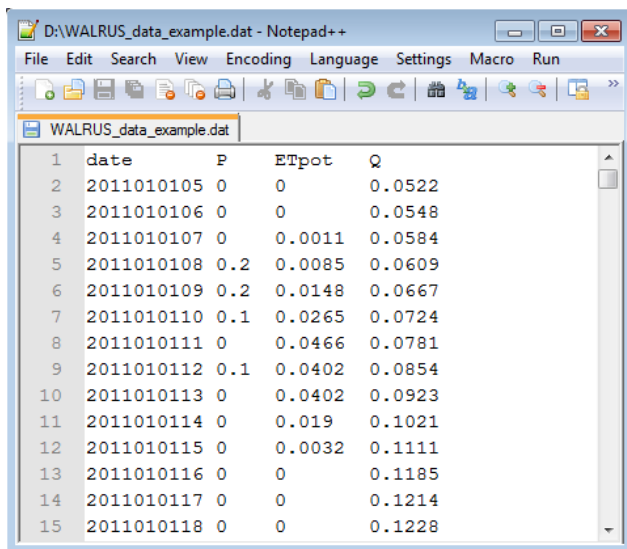
Of the discharge data, only the first observation is used for the simulation (to derive initial values, see Sec. 3.2). If discharge observations are not available at all, the time series could be filled with NA, with an estimate for the start of the modelling period. A few missing data in one of the input time series is not a problem – WALRUS will automatically interpolate the gaps in the `WALRUS_preprocessing` function.

In addition, it is nice (but not necessary) to have:

- an estimate of the percentage of the catchment area covered by surface water ( $a_S$ ),
- an estimate of the characteristic channel depth: how deep are the channels generally incised in the landscape or how deep are the channel bottoms below the land surface ( $c_D$ ),
- a stage-discharge relation of the outlet, so you don't have to calibrate it,
- a time series of the elevation of the outlet weir with respect to the channel bottom (if applicable) and
- groundwater depth time series (to compare model output to observations).

Data time series can also be replaced with simulation functions, for example a Poisson rainfall generator or a





	date	P	ETpot	Q
1	2011010105	0	0	0.0522
2	2011010106	0	0	0.0548
3	2011010107	0	0.0011	0.0584
4	2011010108	0.2	0.0085	0.0609
5	2011010109	0.2	0.0148	0.0667
6	2011010110	0.1	0.0265	0.0724
7	2011010111	0	0.0466	0.0781
8	2011010112	0.1	0.0402	0.0854
9	2011010113	0	0.0402	0.0923
10	2011010114	0	0.019	0.1021
11	2011010115	0	0.0032	0.1111
12	2011010116	0	0	0.1185
13	2011010117	0	0	0.1214
14	2011010118	0	0	0.1228

Figure 5.5: Example of a forcing data file.

(sine) function to approximate evapotranspiration. This is not implemented in the current version of WALRUS.

## 5.6 Preprocessing

If you need other preprocessing, for example a correction for interception on rainfall or a correction for land use on potential evapotranspiration (using a time series of reference evapotranspiration and crop factors; see Sec. 2.12), you should do that before running WALRUS\_preprocessing.

## 5.7 Calibration

To find catchment specific parameter values, you need to calibrate by hand or automatically (see Sec. 3.3). For automatic calibration, you need a calibration criterion, which can be based on discharge and/or groundwater. Note that observed groundwater time series are a combination of slow and quick flow paths, which makes it difficult to compare to the groundwater depth modelled in WALRUS and hampers automated calibration.

Section 7.1 contains a list of things you should check during calibration and some tips to improve the model fit. The figures with model variables and residuals are also helpful in assessing the quality of the output. Section 6.4 contains some calibration examples.

## 5.8 Running and postprocessing

Figure 5.6 shows what your screen will look like when you are running WALRUS in RStudio. The top left corner contains the script from which you are working. The top right corner shows which data and functions are in R's memory at the moment. The bottom left corner shows

the commands that are run and here the water balance is printed when you run WALRUS\_postprocessing. The bottom right corner shows the figures which are plotted during the postprocessing step. You can also change this window to the Packages Window or Help Window to access help for the WALRUS package.

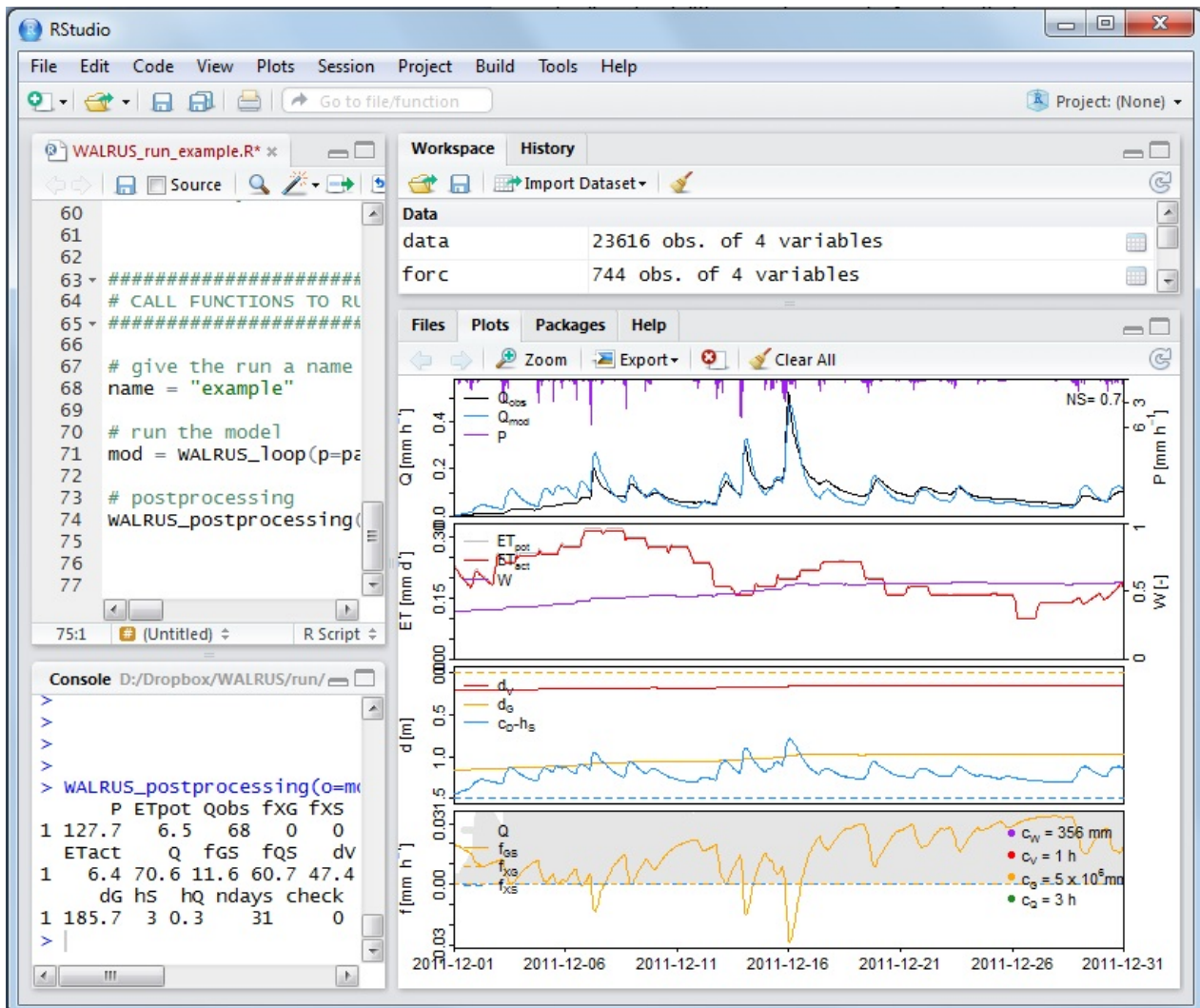


Figure 5.6: Screenshot of WALRUS in R.

## 6 | Examples

In this chapter we show some examples of WALRUS runs, which you can copy, paste and adapt for your own WALRUS model runs. The examples code is printed in the back of this chapter, but can also be downloaded as R scripts from GitHub [www.github.com/ClaudiaBrauer/WALRUS](https://www.github.com/ClaudiaBrauer/WALRUS), in the subfolder called `examples`. We included some comments (behind #, in green) to explain the code.

### 6.1 Basic example

In this basic example all steps required to run WALRUS are performed. You can use this script (see Figure 6.7) as a starting point for your own WALRUS projects. Before running this script, make sure to install WALRUS. The package is loaded in line 7. Make sure that your working directory is set to the right folder (line 11). If the example time series (from the Hupsel Brook catchment) called `PEQ_Hupsel.dat` are located in the subfolder `data` of that working directory, you should be able to run the example without problems.

After running the example script, you should see Figure 6.1 in RStudio and stored in the `figures`-subfolder. The top panel of Fig. 6.1 shows the modelled (blue) and observed (black) discharge. Obviously, these should be as similar as possible, but do realise that observations may contain errors too (disinformation). Purple bars are precipitation observations. The orange line is the modelled groundwater drainage or surface water infiltration and can be viewed as an indicator for the baseflow. In most catchments, the baseflow should touch the streamflow hydrograph when the effect of quickflow after rainfall events decreases to zero. The Nash-Sutcliffe efficiency is also plotted in the top right corner of the plot.

The second panel shows the five-day moving average of the potential evapotranspiration used as input (grey) and the modelled actual evapotranspiration (red), which overlap when there is no evapotranspiration reduction. The flat segments in the curve are caused by the daily cycle: because there is no evapotranspiration at night, the moving average does not change during several hours. The purple line in the second panel represents the wetness index. The wetness index should show seasonal variation, depending on the catchment.

The third panel shows how the levels in the soil reservoir and surface water reservoir change in time and with respect to each other. The dashed blue line represents the channel bottom and the dashed orange line the soil surface. In most catchments, groundwater level (orange) should be above the surface water level (blue) during a large part of the year.

The bottom panel shows some model fluxes, zooming in to groundwater drainage (or surface water infiltration;

orange), seepage (dashed orange) and surface water supply or extraction (dashed blue). The grey background is the modelled discharge. The used parameter values are also plotted in this panel.

After postprocessing (line 42), the water balance is computed (See Eq. 3.3) and saved to file. The balance is also printed in RStudio:

P	ETpot	Qobs	fXG	fXS	ETact	Q	fGS
228.4	15.5	153.9	0	0	15.5	168.7	68.6
fQS	dV	dG	hQ	hS	ndays	check	
98.7	43.5	255.2	0	0.7	62	0	

The fluxes are the sums of all fluxes during the model period. The numbers below `dV`, `hS` and `hQ` are the change in the respective reservoirs over the period. A positive number indicates an increase in the amount of water stored in that reservoir (increase in surface water level ( $h_S$ ) and quickflow reservoir level ( $h_Q$ ), decrease in storage deficit ( $d_V$ ). The number below `dG` indicates the change in groundwater level over the whole period (a positive number meaning an increase of water, so a decrease in groundwater depth). The number of days in the model periods is given as `ndays` and `check` is the rest term of the water budget, which should always be zero.

During postprocessing, several measures of goodness of fit (all based on discharge observations) are computed and stored together with the parameter values. When you open this file in a text editor, you will get Fig. 6.2. The parameters `b`, `psi_ae` and `theta_s` follow from the soil type and the lookup table (Table 2.2). The fraction of the catchment area covered by land (`aG`) is by definition  $1 - a_S$ .

The output file generated during postprocessing contains all model variables. Figure 6.3 shows this data file opened in a text editor. The first column, called `d`, gives the end moment of the time step in seconds since 1 January 1970. The second column `date` gives these moments in `yyyymmddhhmmss` format. `P`, `ETpot`, `Qobs`, `fXG`, `fXS`, `hSmin` and `dG` are the input variables. Some of these input variables may be filled with dummies in case they do not apply to the catchment. Columns `ETact`, `Q`, `fGS` and `fQS` are the fluxes from or between reservoirs. Columns `dV`, `dG`, `hS` and `hQ` are the states of the reservoirs and `dVeq` and `W` are dependent states (which could also be computed from time series of  $d_G$  and  $d_V$ ).

Finally, postprocessing leads to a pdf file with residual plots (Fig. 6.4). The top panel shows the observed precipitation (purple), observed discharge (black) and modelled discharge (blue), just like in Figure 6.1. The second panel shows the residual (modelled discharge minus observed discharge) as a function of time. When the residual is positive, the discharge is overestimated. In most model runs, the largest (absolute) deviations are seen around

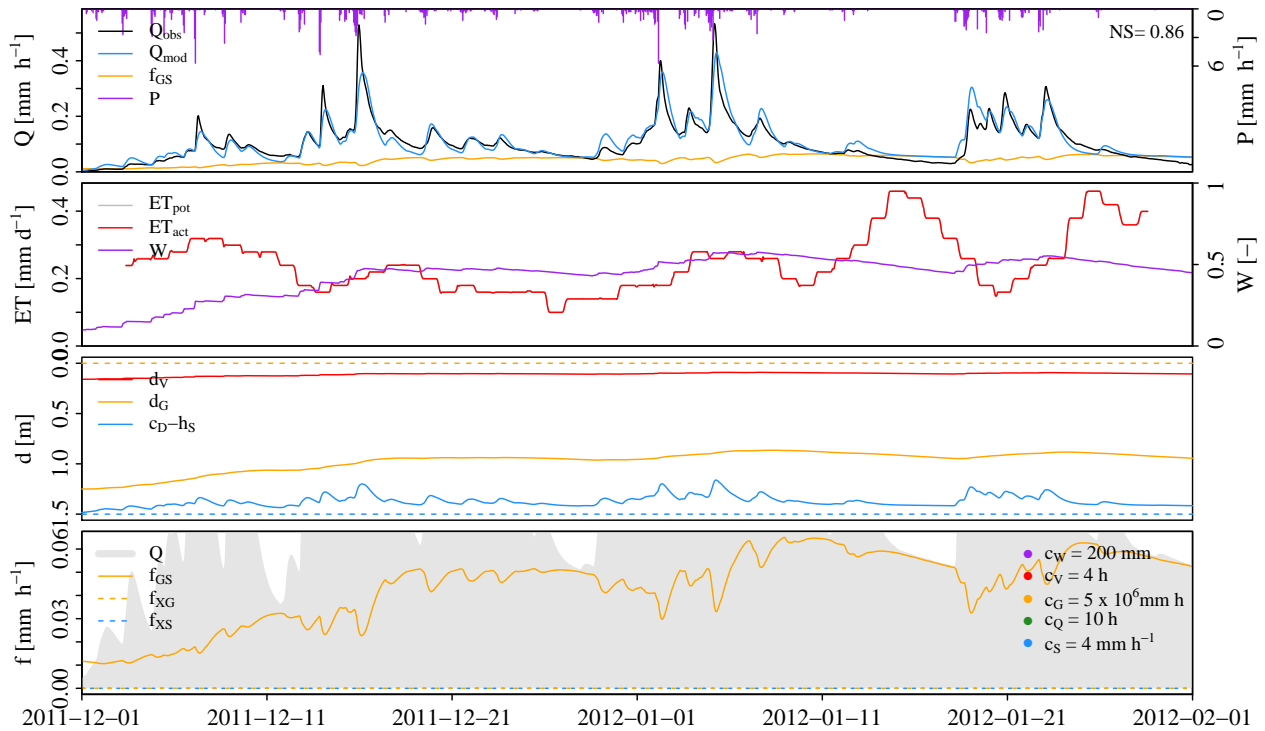


Figure 6.1: Figures generated after running the basic example (Sec. 6.1 and script in Fig. 6.7).

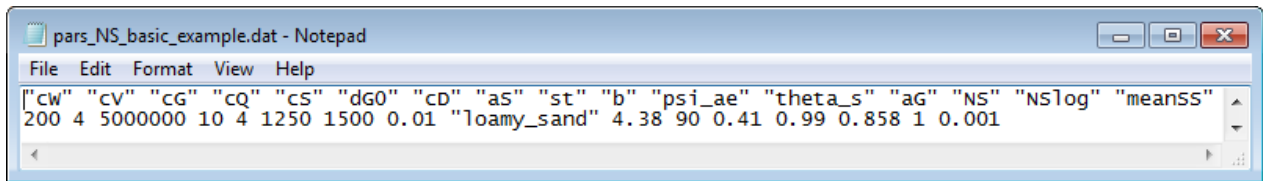


Figure 6.2: File with parameter values and goodness of fit measures generated after running the basic example (Sec. 6.1 and script in Fig. 6.7).

the discharge peaks. The third panel shows the autocorrelation of the residual. If the model error would be random, the autocorrelation would be zero. In reality, the error has some structure and autocorrelation is not zero. When the autocorrelation remains high when the lag time increases, the memory of the error is large. The bottom panel shows the crosscorrelation between the residual and the precipitation time series, which indicates how the error is related to precipitation events. In the example case, the discharge peaks are not sharp enough. Discharge is underestimated directly after the precipitation event and overestimated in the first part of the recession.

Make sure the units of the new stage-discharge relation (lines 15–26) are correct. The argument  $x$  is the stage height in mm (measured from the channel bottom) and the output should be the discharge in  $\text{mm h}^{-1}$ . If your catchment outlet contains a weir of which the elevation changes in time, you can use the argument  $hSmin$  in the stage-discharge relation (and specify  $hSmin$  as a time series in the forcing data set). Whether you need  $hSmin$  in your function or not, be sure to keep the arguments listed in line 15 the same: `function(x, pars, hSmin)`.

## 6.2 Changing default functions

The example in this section shows how one of the four default functions can be changed. Running this code (Fig. 6.8) will change the relation (line 18 in the script), show it on the screen (line 21) and plot the relation (lines 27–30). The resulting screen is shown in Figure 6.5

## 6.3 Changing parameters and initial conditions

The effect of model parameters on the performance is very large. The effect of each parameter on internal model variables can be seen in Figures 2.2 ( $c_W$ ), 2.4 ( $c_V$ ), 2.6 ( $c_G$ ) and 2.7 ( $c_Q$ ). Model parameters can be specified in different ways, as shown in the code in Fig. 6.9).

output\_basic\_example.dat - Notepad

File Edit Format View Help

"d"	"date"	"p"	"ETpot"	"Qobs"	"fxG"	"fXS"	"hSmin"	"Gobs"	"ETact"	"Q"	"fGS"	"fQS"	"dv"	"dveq"	"dG"	"hs"	"hq"	"w"
1322694000	20111130230000	NA	NA	NA	NA	NA	0	NA	0.0044	NA	NA	149.99	149.99	1200	15.98	0	0.1465	
1322697600	20111201000000	0	0	0.0044	0	0	0	0.0044	0.017	0	150.01	149.99	1200	17.25	0	0.1464		
1322701200	20111201010000	0	0	0.0039	0	0	0	0.0049	0.017	0	150.02	149.99	1200	18.45	0	0.1463		
1322704800	20111201020000	0	0	0.0039	0	0	0	0.0055	0.0169	0	150.04	149.99	1200.01	19.59	0	0.1462		
1322708400	20111201030000	0	0	0.0044	0	0	0	0.006	0.0168	0	150.06	149.99	1200.02	20.68	0	0.1461		
1322712000	20111201040000	0	0	0.0044	0	0	0	0.0065	0.0168	0	150.07	150	1200.04	21.71	0	0.146		
1322715600	20111201050000	0	0	0.0044	0	0	0	0.007	0.0167	0	150.09	150	1200.06	22.68	0	0.1459		
1322719200	20111201060000	0.5	0	0.0044	0	0	0	0.0074	0.0166	0	149.68	150	1200.08	24.1	0.07	0.1482		
1322722800	20111201070000	0.3	0	0.0016	0.0044	0	0	0	0.0016	0.0081	0.0165	0.0073	149.44	149.99	1200	25.97	0.11	0.1496
1322726400	20111201080000	0	0	0.0078	0.0044	0	0	0	0.0077	0.0091	0.0164	0.011	149.47	149.96	1199.87	27.79	0.1	0.1494
1322730000	20111201090000	0	0	0.0422	0.0044	0	0	0	0.0419	0.0101	0.0163	0.0099	149.53	149.94	1199.74	29.37	0.09	0.1491
1322733600	20111201100000	0	0	0.0438	0.005	0	0	0	0.0435	0.011	0.0163	0.0089	149.59	149.92	1199.64	30.74	0.08	0.1488
1322737200	20111201110000	0	0	0.0672	0.005	0	0	0	0.0668	0.0117	0.0162	0.008	149.67	149.91	1199.55	31.92	0.07	0.1483
1322740800	20111201120000	0	0	0.025	0.005	0	0	0	0.0248	0.0124	0.0161	0.0072	149.71	149.9	1199.49	32.99	0.06	0.1481
1322744400	20111201130000	0.2	0	0.0094	0.005	0	0	0	0.0093	0.013	0.0161	0.0065	149.57	149.89	1199.45	34.13	0.09	0.1489
1322748000	20111201140000	0	0	0.0031	0.005	0	0	0	0.0031	0.0137	0.016	0.0088	149.58	149.87	1199.37	35.23	0.08	0.1488
1322751600	20111201150000	0.4	0	0.005	0	0	0	0	0.0144	0.016	0.0079	149.26	149.86	1199.3	36.58	0.13	0.1506	
1322755200	20111201160000	0	0	0.0055	0	0	0	0	0.0152	0.0159	0.013	149.28	149.83	1199.15	37.95	0.12	0.1505	
1322758800	20111201170000	0	0	0.0055	0	0	0	0	0.0161	0.0158	0.0117	149.29	149.81	1199.01	39.09	0.11	0.1504	
1322762400	20111201180000	0	0	0.0055	0	0	0	0	0.0168	0.0158	0.0105	149.31	149.78	1198.88	40.04	0.09	0.1503	
1322766000	20111201190000	1	0	0.0061	0	0	0	0	0.0174	0.0157	0.0095	148.47	149.76	1198.76	41.81	0.24	0.155	
1322769600	20111201200000	1.2	0	0.0061	0	0	0	0	0.0186	0.0156	0.0235	147.48	149.7	1198.44	45.07	0.4	0.1607	
1322773200	20111201210000	0.3	0	0.0066	0	0	0	0	0.0208	0.0155	0.0398	147.24	149.6	1197.88	48.81	0.41	0.1621	
1322776800	20111201220000	0.3	0	0.0072	0	0	0	0	0.0235	0.0153	0.0406	147	149.49	1197.29	52.35	0.41	0.1635	
1322780400	20111201230000	0	0	0.0078	0	0	0	0	0.0261	0.0152	0.0413	147.02	149.38	1196.67	55.39	0.37	0.1634	
1322784000	20111202000000	0	0	0.0083	0	0	0	0	0.0284	0.015	0.0372	147.03	149.27	1196.08	57.77	0.33	0.1633	
1322787600	20111202010000	0	0	0.0089	0	0	0	0	0.0302	0.015	0.0334	147.05	149.17	1195.52	59.59	0.3	0.1632	

Figure 6.3: Output file generated after running the basic example (Sec. 6.1 and script in Fig. 6.7).

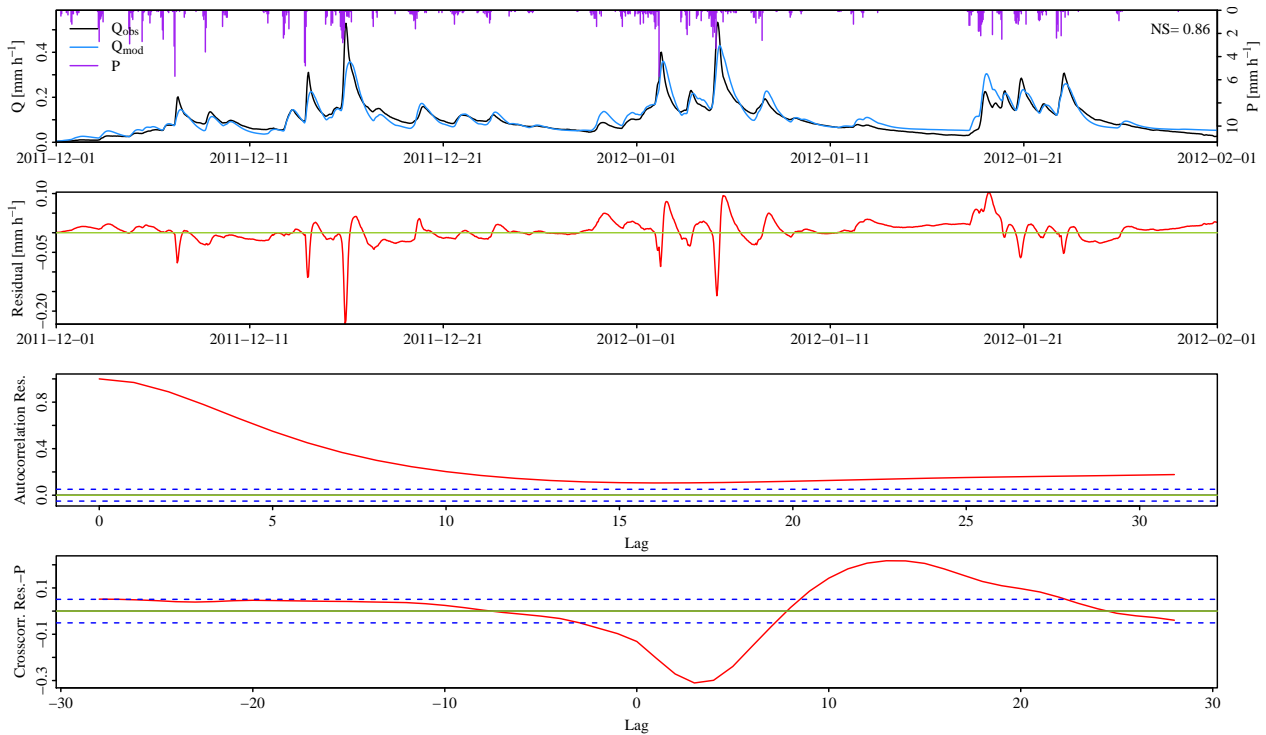


Figure 6.4: Residual plots generated after running the basic example (Sec. 6.1 and script in Fig. 6.7).

The most straightforward way is to specify all model parameters separately (lines 16–17 in the script in Fig. 6.9). You can also read the parameters from file. For example, you could read the output file in which the parameters and the goodness of fit for the basic example are stored (line 25). Reading data files can be especially useful when you are running many parameters sets or to view the results of calibration.

Most numbers in lines 16–17 are model parameters which require calibration:  $cW$ ,  $cV$ ,  $cG$ ,  $cQ$  and  $cS$ .  $cD$ ,  $aS$

and  $st$  are parameters which are usually not calibrated, but estimated based on catchment characteristics. The abbreviation  $st$  stands for soil type and will be used in the `WALRUS_loop`-function to look up  $b$ ,  $\Psi_{ae}$  and  $\theta_s$  in a table (Table 2.2). Initial values for model states can be supplied directly ( $dV0$ ,  $dG0$ ,  $hQ0$ ,  $hS0$ ) or indirectly ( $Gfrac$  and  $Q0$ ). These other options to define initial conditions are explained in Section 3.2



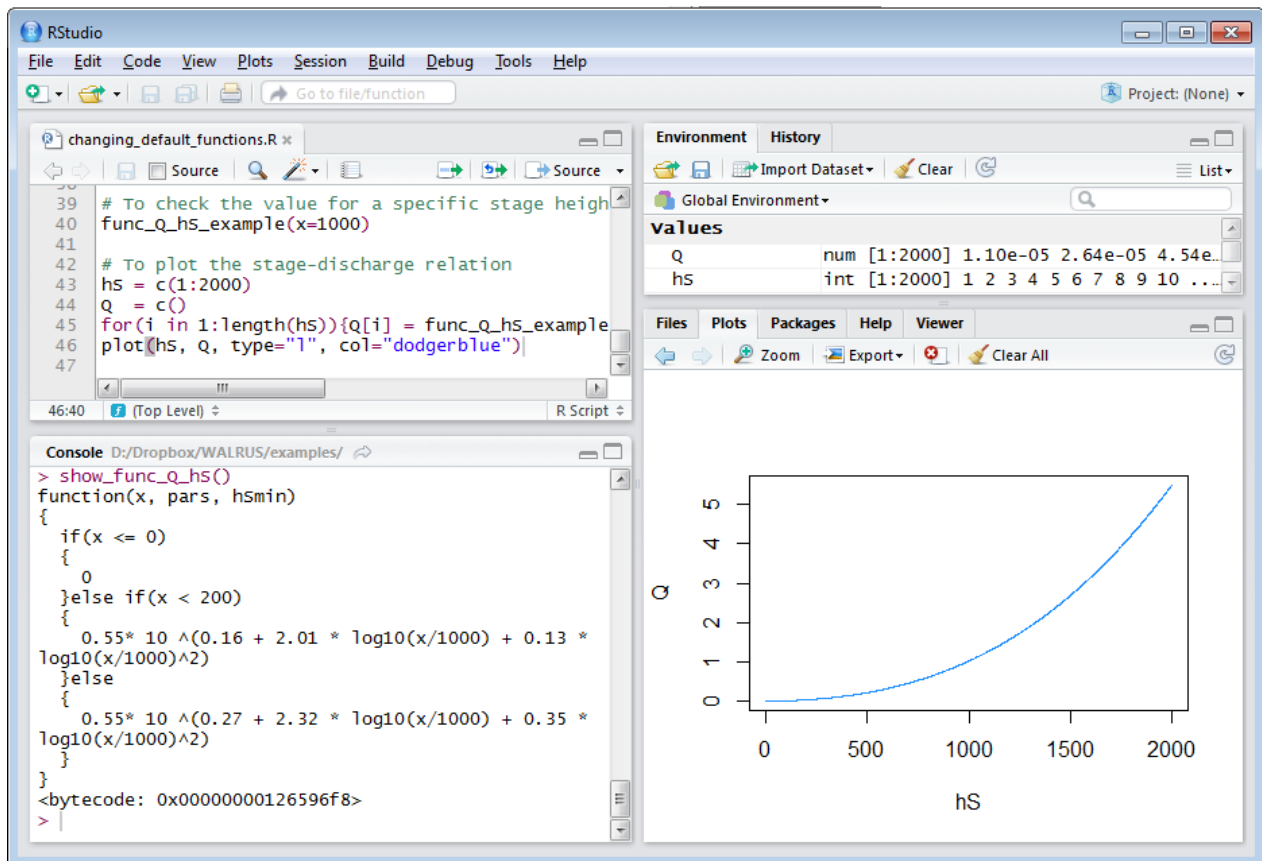


Figure 6.5: Screenshot of the script to change the default stage-discharge relation and the resulting plot and results in RStudio (Sec. 6.3 and script in Fig. 6.8).

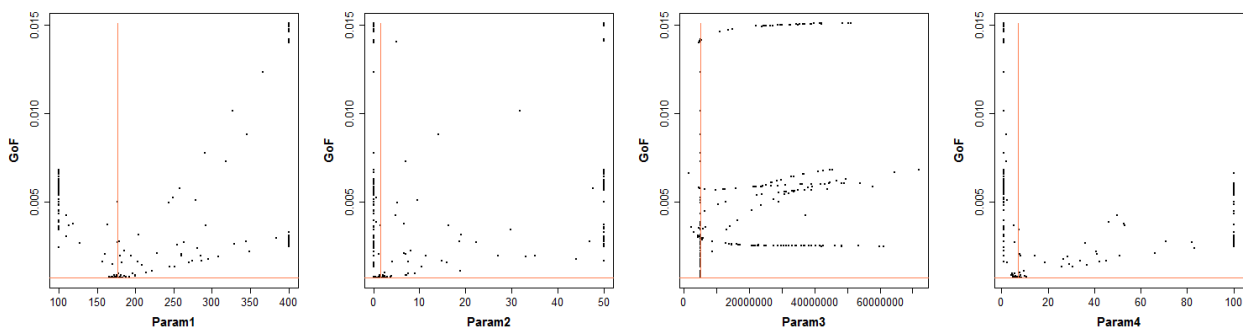


Figure 6.6: Dotty plots showing the relation between parameter value and goodness of fit. These figures are created automatically after calibration with hydroPSO (Sec. 6.4 and script in Fig. 6.10 and 6.11).

## 6.4 Calibration

The script in Figs. 6.10 and 6.11 contain two examples of automatic calibration: one with a particle swarm optimization technique called hydroPSO and one with a Monte Carlo analysis. The first part of the script is similar to the basic example, except line 9, where the package hydroPSO is loaded and line 28, where a special preprocessing step for calibration is performed.

For many calibration techniques, boundaries for parameter values should be given, as well as starting values

for the optimization method. This is done in lines 38–40. In lines 44–50 the model is rewritten, such that the input to the function is a vector with parameter values and the output the goodness of fit. In lines 52–57 WALRUS is run with the starting parameter values, to check if the model runs without errors with the starting parameter set.

Calibration with hydroPSO is done in lines 67–68, in one function called `hydroPSO`. Arguments to this function are the starting parameter set (`par`) and the lower and upper bounds for the parameters (`lower` and `upper`), the rewritten function (`fn`) and a list of control variables

(control). In the list with control variables, the user can specify the number of particles in the swarm (`npart`) and the amount of output shown on the screen during the calibration process (`REPORT`).

The calibration procedure may take a lot of time, depending on the length and temporal resolution of the modelled time series and the number of particles. Running this example (3 months of hourly data with 2 particles) takes about 15 minutes on my computer. The output is stored in several txt files in an automatically generated folder called `PSO.out`: the optimal parameter set `BestParameterSet.txt` being the most important one.

The `hydroPSO` package includes the option to view the output of the calibration. In line 71, the function `plot_results` is called, which creates a new subfolder in the `PSO.out` folder called `pngs`, in which several png files are stored. These Figures visualise the evolution of the parameter values during the calibration procedure and the dependence between parameters. In Figure 6.6 we show one of the `HydroPSO` output Figures. Dotty plots give information about the relation between parameter values and the goodness of fit (Fig. 6.6). Ideally, the points show an optimum in the middle. If the best points are at the edge of the plot, it may mean that the boundaries have not been chosen well or that the parameter compensates for bad values of other parameters. For more information on `HydroPSO` and its output Figures, please consult articles and vignette on [www.rforge.net/hydroPSO/](http://www.rforge.net/hydroPSO/).

In lines 74–81 the model is ran once more with the optimal parameter set. The postprocessing function is run as well (in contrast to running the function through the function defined for calibration purposes in lines 44–50) to make figures and store output data files.

The other calibration technique, the Monte Carlo technique, is quite straightforward, but computationally heavy (10,000 simulations of one year of hourly data take two days on my computer). You may choose to do a Monte Carlo analysis if you want to explore the whole parameter domain and create more extensive dotty plots than in Fig. 6.6. A large number of parameter sets is generated randomly, with numbers varying between set bounds (lines 87–90). The for-loop in lines 96–107 runs over each parameter set, runs the model, and stores the goodness of fit (in this case the sum of squares, *SS*) in a vector (which has been defined before the for-loop in line 93). This vector can be pasted behind the parameter sets and saved to file (line 110). A user can simply take the parameter set belonging to the best model performance or take several good parameter sets (to get an estimate of parameter uncertainty). The data can also be used to investigate the relation between parameters and model performance and dependence between parameters.

```

1  ### Getting started
2
3  # Remove everything from R's memory.
4  rm(list=ls())
5
6  # Load the WALRUS package.
7  library(WALRUS)
8
9  # Change working directory to the folder where data-, figures- and output-subfolders
10 # are located.
11 setwd("D:/Dropbox/WALRUS/runs/examples/")
12
13 ### Data
14
15 # Read daily or hourly precipitation, potential evapotranspiration and discharge data.
16 data = read.table("data/PEQ_Hupsel.dat", header=TRUE)
17
18 # Specify which period of the total data set you want to use as forcing data.
19 # Use the same date format as in data file (for example yyyyymmddhh).
20 forc = WALRUS_selectdates("data", 2011120000, 2012020000)
21
22 # Preprocessing forcing and specifying for which moments output should be stored.
23 # The argument dt is time step size used in the output data file (in hours).
24 WALRUS_preprocessing(f=forc, dt=1)
25
26 ### Parameters and initial values
27
28 # Define the parameters (cW, cV, cG, cQ, cS), initial conditions (dG0) and
29 # catchment characteristics (cD, aS, soil type).
30 pars = data.frame(cW=200, cV=4, cG=5e6, cQ=10, cS=4,
31                   dG0=1250, cD=1500, aS=0.01, st="loamy_sand")
32
33 ### Call functions to run model
34
35 # Run the model.
36 mod = WALRUS_loop(pars=pars)
37
38 # Give the run a logical name. This will be used for the output data files and figures.
39 name = "basic_example"
40
41 # Postprocessing: create datafiles and show figures.
42 WALRUS_postprocessing(o=mod, pars=pars, n=name)

```

Figure 6.7: R script (basic\_example.R) for the basic example (Sec. 6.1).



```

1  ##### Change Q-h-relation
2
3  # Build a function in which the discharge is computed as a function of stage height.
4  func_Q_hS_example = function(x, pars, hSmin)
5  {
6      if(x <= 0)
7      {
8          0
9      } else if(x < 200)
10     {
11         0.55* 10 ^ (0.16 + 2.01 * log10(x/1000) + 0.13 * log10(x/1000)^2)
12     } else {
13         0.55* 10 ^ (0.27 + 2.32 * log10(x/1000) + 0.35 * log10(x/1000)^2)
14     }
15 }
16
17 # Then set this function as the current stage-discharge relation.
18 set_func_Q_hS(func_Q_hS_example)
19
20 # And to check if you did it right:
21 show_func_Q_hS()
22
23 # To check the value for a specific stage height manually:
24 func_Q_hS_example(x=1000)
25
26 # To plot the stage-discharge relation
27 hS = c(1:2000)
28 Q = c()
29 for(i in 1:length(hS)){Q[i] = func_Q_hS_example(hS[i])}
30 plot(hS, Q, type="l", col="dodgerblue")

```

Figure 6.8: Code from the example (changing\_default\_functions.R) on how to change the stage-discharge relation (Sec. 6.2).

```

1  ### Getting started
2
3  # Remove everything from R's memory.
4  rm(list=ls())
5
6  # Load the WALRUS package.
7  library(WALRUS)
8
9  # Change working directory.
10 setwd("D:/Dropbox/WALRUS/examples/")
11
12
13 ### Parameters
14
15 # Option 1: define the parameters and initial conditions manually.
16 pars = data.frame(cW=500, cV=4, cG=5e6, cQ=1, cS=5,
17                  dG0=1000, cD=1500, aS=0.01, st="loamy_sand")
18 # The options for st (soil type) are:
19 # "sand", "loamy_sand", "sandy_loam", "silt_loam", "loam", "sandy_clay_loam",
20 # "silt_clay_loam", "clay_loam", "sandy_clay", "silty_clay", "clay", "cal_H", "cal_C").
21 # The values for "cal_H" and "cal_C" were obtained by fitting on observations in
22 # the Hupsel Brook catchment and Cabauw polder.
23
24 # Option 2: read a parameter set from file.
25 pars = read.table("output/pars_NS_basic_example.dat", header=TRUE)
26
27
28 ### Initial conditions
29
30 # The initial conditions can be set in four ways.
31 # Option 1: set the groundwater depth at t=0
32 # (in this case at 1000 mm) with the argument dG0=1000.
33 # This option is also used in the basic example.
34 pars = data.frame(cW=500, cV=4, cG=5e6, cQ=1, cS=5,
35                  dG0=1000, cD=1500, aS=0.01, st="loamy_sand")
36
37 # Option 2: specify the fraction of discharge which originates from groundwater
38 # (in this case 80%), with the argument Gfrac=0.8.
39 pars = data.frame(cW=500, cV=4, cG=5e6, cQ=1, cS=5,
40                  Gfrac=0.8, cD=1500, aS=0.01, st="loamy_sand")
41
42 # Option 3: without specifying either dG0 or Gfrac.
43 # By default, all discharge will be assumed to originate from groundwater.
44 pars = data.frame(cW=500, cV=4, cG=5e6, cQ=1, cS=5,
45                  cD=1500, aS=0.01, st="loamy_sand")
46
47 # Option 4: without specifying either dG0 or Gfrac, but by using a warming-up period.
48 # During this period, the model is run, but the output is not stored and
49 # used to compute goodness of fit.
50 # The warming-up period is specified with the forcing time series in hours.
51 # For example, if you want a warming-up period of one month:
52 data = read.table("data/PEQ_Hupsel.dat", header=TRUE)      # read data
53 forc = WALRUS_selectdates("data", 2011120000, 2012020000)  # select specific period
54 forc$warm = 24*30                                           # define warming-up period
55 WALRUS_preprocessing(f=forc, dt=1)                          # run preprocessing function

```

Figure 6.9: Code from the example (changing\_parameters.R) on how to change parameters and initial conditions (Sec. 6.3).

```

1  ### Getting started
2
3  # Remove everything from R's memory.
4  rm(list=ls())
5
6  # Load the WALRUS package and the HydroPSO package for the particle swarm
7  # optimisation technique by Zambrano-Bigarini and Rojas (2013).
8  library(WALRUS)
9  library(hydroPSO)
10
11 # Change working directory to the folder where the subfolders "data", "figures" and
12 # "output" are located.
13 setwd("D:/Dropbox/WALRUS/examples/")
14
15
16 ### Forcing
17
18 # Read daily or hourly precipitation, potential evapotranspiration and discharge data.
19 data = read.table("data/PEQ_Hupsel.dat", header=TRUE)
20
21 # Specify which period of the total data set you want to use as forcing data.
22 # Use the same date format as in data file (for example yyyyymmddhh).
23 forc = WALRUS_selectdates("data", 2011120000, 2012020000)
24
25 # Preprocessing forcing and output data file. The argument dt is time step size used
26 # in output data file (in hours).
27 WALRUS_preprocessing(f=forc, dt=1)
28 WALRUS_preprocessing_calibration()
29
30
31 ### Prepare calibration
32
33 # Give the run a logical name.
34 name = "calibration_example"
35
36 # Define initial and boundary values for the parameters (and/or initial conditions)
37 # you want to calibrate.
38 par_start = c(200, 4, 5e6, 10)
39 par_LB = c(100, 0.1, 0.1e6, 1)
40 par_UB = c(400, 50, 100e6, 100)
41
42 # Rewrite the model such that the output is the goodness of fit (evaluation of the
43 # calibration criterion).
44 WALRUS_for_optim = function(par)
45 {
46   fit_pars = data.frame(cW=par[1], cV=par[2], cG=par[3], cQ=par[4], cS=4,
47                         dG0=1200, cD=1500, aS=0.01, st="loamy_sand")
48   mod = WALRUS_loop(pars=fit_pars)
49   return(mean((Qobs_forNS-mod$Q)^2, na.rm=TRUE))
50 }
51
52 # Test if the starting model parameters yield reasonable results.
53 fit_pars = data.frame(cW=par_start[1], cV=par_start[2], cG=par_start[3],
54                      cQ=par_start[4], cS=4,
55                      dG0=1200, cD=1500, aS=0.01, st="loamy_sand")
56 mod = WALRUS_loop(pars=fit_pars)
57 WALRUS_postprocessing(o=mod, pars=fit_pars, n=name)

```

Figure 6.10: First half of the code from the example (calibration.R) on calibration (Sec. 6.4).

```

60 ##### Calibration
61
62 # The particle swarm optimization.
63 # REPORT determines the amount of information shown on the screen during calibration.
64 # Increasing REPORT will give more information.
65 # npart is the number of particles.
66 # Increasing npart will lead to better results, but longer runtimes.
67 cal = hydroPSO(par=par_start, lower=par_LB, upper=par_UB, fn=WALRUS_for_optim,
68               control=list(verbose=TRUE, REPORT=1, npart=2))
69
70 # Make figures of the calibration results (standard output from the hydroPSO-package).
71 plot_results(MinMax="min", do.png=TRUE)
72
73 # Load the optimal parameter values found in the calibration procedure
74 PSO_pars = read.table("PSO.out/BestParameterSet.txt", header=TRUE)
75 opt_pars = data.frame(cW=PSO_pars$Param1, cV=PSO_pars$Param2, cG=PSO_pars$Param3,
76                      cQ=PSO_pars$Param4, cS=4,
77                      dG0=1200, cD=1500, aS=0.01, st="loamy_sand")
78
79 # Run once more with best parameters to create data files and plot figures.
80 mod = WALRUS_loop(pars=opt_pars)
81 WALRUS_postprocessing(o=mod, pars=opt_pars, n=name)
82
83
84 ##### Monte Carlo calibration
85
86 # Make many (in this example 10000) random parameter sets within certain limits.
87 cW = runif(10000, min=100, max=400 )
88 cV = runif(10000, min=0.1, max=50 )
89 cG = runif(10000, min=1e5, max=1e8 )
90 cQ = runif(10000, min=1 , max=100 )
91
92 # Make an empty vector to store mean sums of squares during for-loop.
93 SS = c()
94
95 # Run a for-loop over all parameter sets and run WALRUS in every iteration.
96 for(i in 1:10000)
97 {
98   # look up the parameter set for this iteration
99   parameters = data.frame(cW=cW[i], cV=cV[i], cG=cG[i], cQ=cQ[i], cS=5,
100                          dG0=1000, cD=1500, aS=0.01, st="loamy_sand")
101
102   # run WALRUS
103   modeled = WALRUS_loop(pars=parameters)
104
105   # Compute and store the mean sum of squares.
106   SS[i] = mean((Qobs_forNS-modeled$Q)^2)
107 }
108
109 # Write the results to file: parameter values and belonging sum of squares.
110 write.table(data.frame(cbind(cW, cV, cG, cQ, SS)), "par_SS_MC.dat")

```

Figure 6.11: Second half of the code from the example (calibration.R) on calibration (Sec. 6.4).

This section contains some frequently asked questions and points of attention. In particular, there are certain things you should check during calibration to verify that the model is producing realistic results (and tips how to solve problems if it isn't). Figure 7.1 show an example of WALRUS output with unrealistic elements. This Chapter will grow in time – experience from different users is welcome so we can improve this manual.

## 7.1 Things to check during calibration

### Wetness index variation

Is the wetness index (divider between quickflow and slow flow) varying enough between seasons and after events? Is it what you expect for this catchment?

If  $W$  is always close to zero (yielding only slow flow), you can increase  $c_W$  to increase quickflow contribution (see Fig. 2.2). Also check if the channel depth  $c_D$  is correct. If you overestimate the channel depth, groundwater will become too deep.

If  $W$  is always close to one (yielding only quickflow), you can decrease  $c_W$  to decrease quickflow contribution (see Fig. 2.2).

### Baseflow contribution

Is the baseflow  $f_{GS}$  a large enough portion of the total discharge? In general, it is good to try to get the baseflow to touch the discharge curve.

If  $f_{GS}$  is almost zero or negative, you can decrease  $c_G$  to increase the reaction speed of the slow flow reservoir (and  $f_{GS}$ , see Fig. 2.6) or decrease  $c_W$  to increase the slow flow contribution.

### Discharge peak shapes

Do the discharge peaks have the right shape?

If the peaks are too sharp, decrease  $c_Q$  to get a faster response of the quickflow reservoir (see Fig. 2.7). You also may have underestimated the fraction of the catchment area covered by surface water ( $a_S$ ).

If the peaks are not sharp enough, increase  $c_Q$  to get a slower response of the quickflow reservoir (see Fig. 2.7). You also may want to check the contribution of the quickflow reservoir (Section 7.1). You also may have overestimated the fraction of the catchment area covered by surface water ( $a_S$ ).

### Difference between groundwater and surface water levels

Is the difference between groundwater levels and surface water levels what you expect for your catchment? In most

catchments, the groundwater levels only drop below the surface water levels in small periods in summer and during discharge peaks.

If the groundwater level is too often below the surface water level, you can decrease  $c_W$  to increase the slow flow contribution (see Fig. 2.2). You can also check if the channel depth  $c_D$  is correct. If you overestimate the channel depth, groundwater will become too deep.

### Variation in surface water level

Is the surface water level varying as much as you expect for this catchment?

If the surface water level is not dynamic enough, you may have overestimated  $c_S$ . The parameter  $c_S$  represents the bankfull discharge (in  $\text{mm h}^{-1}$ ), so if this value is too large, your river is too wide and the discharge will be spread out over a large streambed, leading to low surface water levels.

## 7.2 Questions

### Can R/WALRUS only read .dat data files?

The data file can be in any format R can read (.txt, .csv, etc.). To read file formats in which the columns are separated with a character, add the `sep`-argument to the `read.table`-function, specifying the separator. For example, to read a .csv file with separator ; , replace line 16 in the basic example (Fig. 6.7) with

```
data = read.table("data/PEQ_Hupsel.csv",
  header=TRUE, sep=";")
```

### How can I add a constant seepage flux?

After line 20 in the basic example (Fig. 6.7), in which the forcing data frame is made, you can add

```
forc$fGS = 0.1
```

to add one value of the seepage flux (in this case 0.1 mm per timestep) to the forcing data frame. When the pre-processing function is called (line 24), this flux is used for every time step.

## 7.3 Error messages

### Date-time format

Make sure tht you write the date-time formats as `yyyymmdd`, `yyyymmddhh` or `yyyymmddhhmm`, without dashes and spaces and without the minutes and seconds.

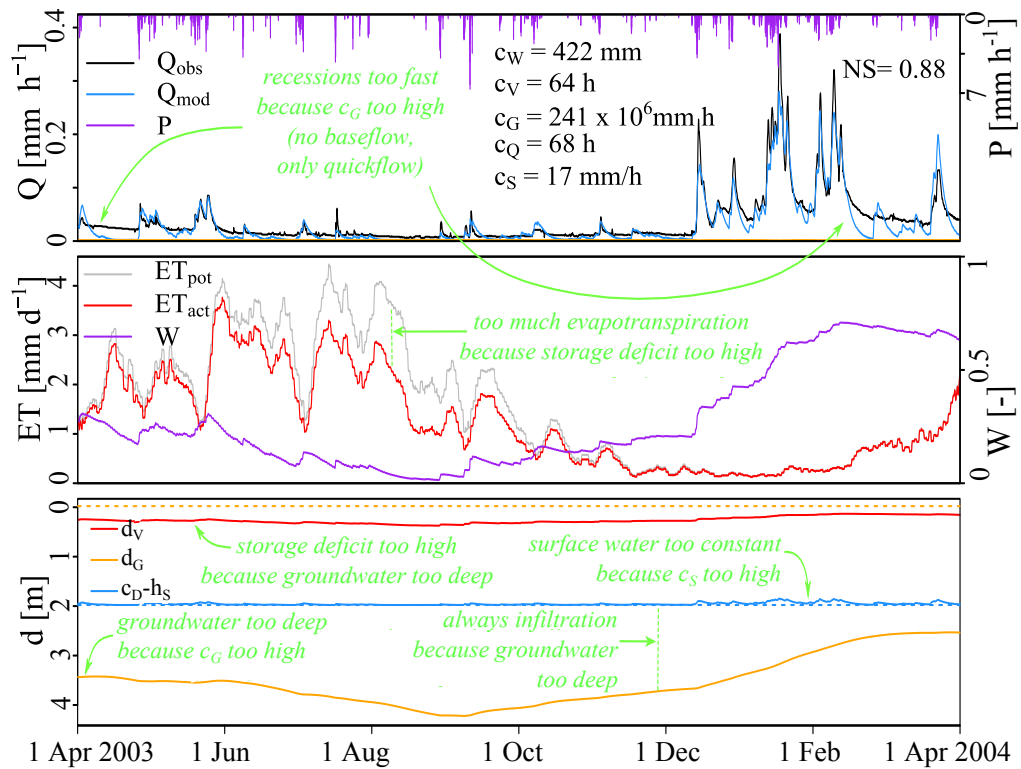


Figure 7.1: Evaluation of the output after calibration. The green text shows which elements are considered unrealistic for this catchment and what (probably) causes these issues (based on Ten Broek, 2014).

### Stage-discharge relation error

If you get this error message:

```
Error in if (is.na(f.upper))
  stop("f.upper = f(upper) is NA") :
  argument is of length zero
```

the initial surface water level cannot be computed from the stage-discharge relation. If you use the default stage-discharge relation, make sure you specify  $c_S$ . If you specified a stage-discharge relation yourself, make sure that the units of the stage-discharge relation are correct ( $Q$  in  $\text{mm h}^{-1}$  and  $h_S$  in mm).

# Bibliography

- Beven, K. J., Kirkby, M. J., 1979. A physically based, variable contributing area model of basin hydrology. *Hydrolog. Sci. J.* 24 (1), 43–69.
- Brauer, C. C., Teuling, A. J., Overeem, A., van der Velde, Y., Hazenberg, P., Warmerdam, P. M. M., Uijlenhoet, R., 2011. Anatomy of extraordinary rainfall and flash flood in a Dutch lowland catchment. *Hydrol. Earth Syst. Sci.* 15, 1991–2005.
- Brauer, C. C., Teuling, A. J., Torfs, P. J. J. F., Uijlenhoet, R., 2014a. The Wageningen Lowland Runoff Simulator (WALRUS): a lumped rainfall-runoff model for catchments with shallow groundwater. *Geosci. Model Dev.* 7, 2313–2332.
- Brauer, C. C., Torfs, P. J. J. F., Teuling, A. J., Uijlenhoet, R., 2014b. The Wageningen Lowland Runoff Simulator (WALRUS): application to the Hupsel Brook catchment and Cabauw polder. *Hydrol. Earth Syst. Sci.* 18, 4007–4028.
- Brooks, R. H., Corey, A. . T., 1964. Hydraulic properties of porous media. *Hydrology Paper* 3, Colorado State University, Fort Collins, CO, 27 pp.
- Clapp, R. B., Hornberger, G. M., 1978. Empirical equations for some hydraulic properties. *Water Resour. Res.* 14, 601–604.
- Cosby, B. J., Hornberger, G. M., Clapp, R. B., Ginn, T. R., 1984. A statistical exploration of the relationships of soil moisture characteristics to the physical properties of soils. *Water Resour. Res.* 20 (6), 682–690.
- Gillham, R. W., 1984. The capillary fringe and its effect on water-table response. *J. Hydrol.* 67 (1), 307–324.
- Kustas, W. P., Rango, A., Uijlenhoet, R., 1994. A simple energy budget algorithm for the Snowmelt Runoff Model. *Water Resour. Res.* 30, 1515–1527.
- Manning, R., 1889. On the flow of water in open channels and pipes. *Trans. Inst. Civ. Eng. Ireland* 20, 161–207.
- Seibert, J., 1997. Estimation of parameter uncertainty in the HBV model. *Nordic Hydrol.* 28, 247–262.
- Ten Broek, J. M., 2014. Coupling WALRUS to SOBEK: Wageningen Lowland Runoff Simulator to 1D open water model. Internship report (at Water Board Rijn and IJssel), Wageningen University.
- Teuling, A. J., Troch, P. A., 2005. Improved understanding of soil moisture variability dynamics. *Geophys. Res. Lett.* 32 (5), L05404.
- Torfs, P. J. J. F., Brauer, C. C., 2015. A (very) short introduction to R. Contributed to R-project. URL [cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf](http://cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf)
- Van Genuchten, 1980. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Sci. Soc. Am. J.* 44, 892–898.
- Wendt, D. E., 2015. Snow hydrology in the Netherlands: Developing snowmelt algorithms for Dutch regional water management modules. Internship report (at HKV), Wageningen University.
- Zambrano-Bigarini, M., Rojas, R., 2013. A model-independent particle swarm optimisation software for model calibration. *Environ. Modell. Softw.* 43, 5–25.





# Index

baseflow, 31

calibration, 11, 19, 24, 31

capillary rise, 6, 7

catchment size, 10

drainage, 7

equilibrium storage deficit, 6

error messages, 31

evapotranspiration, 5, 21

evapotranspiration reduction, 5

figures, 15, 21

file structure, 18

flooding, 9

fluxes, 3

forcing, 11, 15, 18

goodness of fit, 21

graphical user interface, 16

groundwater abstraction, 9

groundwater depth, 31

groundwater drainage, 7, 31

help, 18

hydraulic model, 10

if-statements, 11

initial conditions, 11, 22

input, 15

integration scheme, 12

interception, 10

model equations, 3, 4, 15

model parameters, 3, 11

model states, 3

model structure, 3

model variables, 3, 4

output, 15, 21

package, 15, 17

parameters, 22

percolation, 6, 7

ponding, 9

postprocessing, 15, 19

preprocessing, 15, 19

quickflow, 8, 31

R, 15

reservoir levels, 21

reservoirs, 3

residuals, 21

seepage, 9

snow, 10

soil moisture profile, 6

stage-discharge relation, 22, 31

storage deficit, 6

surface water, 8

surface water extraction, 9

surface water infiltration, 7

surface water level, 31

surface water supply, 9

time step, 12

urban areas, 10

vegetation, 9

water balance, 12, 21

wetness index, 5, 31