LISFLOOD: a GIS-based distributed model for river basin scale water balance and flood simulation

J. M. Van Der Knijff *, J. Younis *, A. P. J. De Roo *

* Land Management & Natural Hazards Unit, Institute for Environment and Sustainability, DG Joint Research Centre, European Commission, Italy

First Published on: 24 November 2008
Research Article

LISFLOOD: a GIS-based distributed model for river basin scale water balance and flood simulation

J. M. VAN DER KNIJFF, J. YOUNIS and A. P. J. DE ROO*
Land Management & Natural Hazards Unit, Institute for Environment and Sustainability, DG Joint Research Centre, European Commission, Italy

(Received 30 July; in final form 13 September 2008)

In this paper we describe the spatially distributed LISFLOOD model, which is a hydrological model specifically developed for the simulation of hydrological processes in large European river basins. The model was designed to make the best possible use of existing data sets on soils, land cover, topography and meteorology. We give a detailed description of the simulation of hydrological processes in LISFLOOD, and discuss how the model is parameterized. We also describe how the model was implemented technically using a combination of the PCRaster GIS system and the Python programming language, and discuss the management of in- and output data. Finally, we review some recent applications of LISFLOOD, and we present a case study for the Elbe river.

Keywords: LISFLOOD; PCRaster; Rainfall-runoff models; Floods

1. Introduction

LISFLOOD is a GIS-based hydrological rainfall-runoff-routing model that is capable of simulating the hydrological processes that occur in a catchment. The specific development objective was to produce a tool that can be used in large and transnational catchments for a variety of applications, including flood forecasting, and assessing the effects of river regulation measures, land-use change and climate change. Although a wide variety of existing hydrological models are available that are suitable for each of these individual tasks, few single models are capable of doing all these jobs. For example, the Swedish HBV hydrology model (Hydrologiska Byråns Vattenbalansmodell) (e.g. Lindström et al. 1997) is a rainfall-runoff model with appropriate process descriptions for our needs, but it lacks a spatially distributed river routing component. MIKE-SHE (DHI 2000) is a very good physically-based model, but it cannot be used for larger river basins. MIKE-11 (Havno et al. 1995) is better suited in this respect, but its rainfall-runoff component is not quite sophisticated enough for our purposes. HEC-RAS (Brunner 2008) is limited to river routing only and does not contain a rainfall-runoff component at all. TOPKAPI (Ciarapica and Todini 2002) is a river basin model that extends the classic TOPMODEL (Beven and Kirkby 1979) approach. Its current range of application fields shows some overlap with LISFLOOD; however, TOPKAPI was applied to and tested for smaller river basins only when the development of LISFLOOD started. The American HEC-HMS (Scharffenberg and Fleming 2008) is a semi-lumped model.

*Corresponding author. Email: ad.de-roo@jrc.it
Our objective requires a model that is spatially distributed and—at least to a certain extent—physically-based. Also, the focus of our work is on European catchments. Since several (spatial) databases exist that contain pan-European information on soils (King et al. 1997, Wösten et al. 1999), land cover (CEC 1993), topography (Hiederer and de Roo 2003) and meteorology (Rijks et al. 1998), it would be advantageous to have a model that makes the best possible use of these data. Finally, the wide scope of our objective implies that changes and extensions to the model will be required from time to time. Therefore, it is essential to have a model code that can be easily maintained and modified. LISFLOOD has been specifically developed to satisfy these requirements. In parallel to LISFLOOD, a separate flood inundation model called LISFLOOD-FP has been developed as well (Bates and de Roo 2000). To avoid any confusion, we would like to stress that both are different (although complementary) models. LISFLOOD-FP will not be discussed in this paper.

Some examples of recent applications of LISFLOOD are given in Feyen et al. (2007), Feyen et al. (2008), Gouweleeuw et al. (2004), Dankers et al. (2007), Thielen et al. (2008) and Younis et al. (2008a, b). These papers include a brief description of the model only. A paper by de Roo et al. (2000) describes an earlier version of the model. Considerable changes have been incorporated into the model since that paper was published. The aim of the current paper is to provide an up-to-date description that reflects the current state of the model. We do this by first outlining in section 2 the general characteristics of the model, followed by a description of the individual processes that are included. In section 3 we provide an overview of the methods and data sources that are used to parameterize the model. In section 4 we explain how we implemented the model using a combination of the PCRaster Dynamic Modelling language (Wesseling et al. 1996, Karssenberg 2002) and the Python scripting language (Python 2008). We also explain here why we decided on such an approach. Section 5 discusses the management of LISFLOOD’s in- and output data. Finally, in section 6 we give an overview of some recent applications of the model, and present a new case study. We end with a concluding section.

2. Simulation of hydrological processes

LISFLOOD is a spatially distributed, grid-based rainfall-runoff and channel routing model. It can run using any desired time interval, on any grid size. The model is typically run using a daily time interval to simulate the long-term catchment water balance, whereas smaller intervals (e.g. hourly) are better suited to modelling individual flood events. Both can be combined as well. For instance, the state variables at the end of a (daily) water balance run can be used to provide the initial conditions for an (hourly) flood run. The model does not impose any limitations on the grid resolution that is used. However, its separation between runoff-generating and channel routing processes would be poorly represented at very low pixel resolutions. Since LISFLOOD has been primarily developed for the simulation of large river basins, small-scale processes are often simulated in a simplified way. Because of this, there would be little benefit in using very high resolutions either. We would therefore recommend using the model at grid resolutions within the range of 100–10 km. Most current applications of the model have employed grid resolutions of 1 or 5 km. Figure 1 gives an overview of the structure of LISFLOOD. As the figure shows, the model is made up of a two-layer soil water balance sub-model, sub-models for the simulation of groundwater and subsurface flow (using two parallel
interconnected linear reservoirs), a sub-model for the routing of surface runoff to the nearest river channel, and a sub-model for the routing of channel flow (not shown in the figure).

The processes that are simulated by the model include snow melt (not shown in figure 1), infiltration, interception of rainfall, leaf drainage, evaporation and water uptake by vegetation, surface runoff, preferential flow (bypass of soil layer), exchange of soil moisture between the two soil layers and drainage to the groundwater, sub-surface and groundwater flow, and flow through river channels. Upward vertical soil moisture and groundwater flow (capillary rise) are not simulated, and neither are deep groundwater systems. This poses some limitations on the use of LISFLOOD in areas that are either very dry or have a hydrology that is heavily influenced by deep groundwater, or combinations of both.

Most hydrological processes can be modelled in different ways, and process descriptions may be anything within the range between simple empirical ‘black box’ relations and fully ‘physically based’ approaches (which can be both numerically complex and computationally demanding). As stated already in the introductory section, our objective requires process descriptions that are physically based to some extent. At the same time, in order to be of any practical use, the model should be computationally efficient to a sufficient degree. Moreover, the often approximate
nature of data derived from large-scale datasets—such as the pan-European databases mentioned in the introduction—would not support an approach that is fully physically based. Finally, limitations of the more physically based approaches have been discussed at length in the more recent literature (see e.g. Beven 2001 for an overview). For LISFLOOD we have aimed to select process descriptions that make the best use of available prior data – thus reducing the number of calibration parameters, but we have tried to avoid process descriptions that are overly complex, computationally demanding or irrelevant at the scale of large catchments. In the following we describe the individual processes in more detail.

2.1 Meteorological forcing

LISFLOOD is driven by the following meteorological variables: precipitation intensity, $P$ (mm day$^{-1}$), potential (reference) evapotranspiration rate of a closed canopy, $ET_0$ (mm day$^{-1}$), potential (reference) evaporation rate from a bare soil surface, $ES_0$ (mm day$^{-1}$), potential evaporation rate from an open water surface, $EW_0$ (mm day$^{-1}$), and average 24-hour temperature, $T_{avg}$ ($^\circ$C). $ET_0$, $ES_0$ and $EW_0$ are all calculated outside the model, and a separate pre-processing application that calculates these variables from standard meteorological observations is available as a companion to the model. LISFLOOD always expects these input variables in the units as given above, irrespective of the actual time step used. In other words: even if the model is run on an hourly time step, precipitation must be provided as an intensity with units (mm day$^{-1}$). Note that in the remainder of the description that follows, all rate variables are expressed in (mm) per time step, unless stated otherwise.

2.2 Snow and frost

If the average daily temperature is below 1$^\circ$C, all precipitation is assumed to be snow. A snow correction factor can be applied to correct for undercatch of snow precipitation. Unlike rain, snow accumulates on the soil surface until it melts. Rates of snowmelt can be estimated by simulating the full surface radiation balance. However, a comparative study of different snowmelt models by the World Meteorological Organization did not demonstrate such models to be superior to much simpler modelling approaches that are based on temperature indices (WMO 1986). Since radiation balance models are rather data-demanding (both in terms of parameters that need to be estimated as well in meteorological input data), LISFLOOD uses the following simple degree-day factor equation instead (Speers et al. 1979, cited in Young 1985):

$$M = C_m(1 + 0.01 \cdot R \Delta t)(T_{avg} - T_m) \cdot \Delta t$$

where $M$ is the rate of snowmelt (mm), $C_m$ is a degree-day factor (mm $^\circ$C$^{-1}$ day$^{-1}$), $R$ is the rainfall intensity (mm day$^{-1}$), $\Delta t$ is the time interval (days), $T_{avg}$ is the average 24-hour temperature ($^\circ$C), and $T_m$ is the temperature above which snowmelt occurs ($^\circ$C). The equation takes into account accelerated snowmelt when it is raining. For large pixel sizes, there may be considerable sub-pixel heterogeneity in snow accumulation and melt, which is a particular problem if there are large elevation differences within a pixel. Because of this, melt and accumulation of snow are modelled separately for three separate elevation zones, which are defined at the sub-pixel level.
When the soil is frozen, this affects the hydrological processes occurring near its surface. In LISFLOOD it is assumed that evaporation from the soil surface, water uptake by vegetation, infiltration, and flow of moisture through the soil matrix are all reduced to zero. To establish whether the soil surface is frozen or not, a frost index $F$ is calculated (re-written from Molnau and Bissell 1983, cited in Maidment 1993):

$$F(t) = F(t-1) - (1 - A_f) F \Delta t - T_{avg} e^{-0.04Kd_s/wes} \Delta t$$

where $F$ is expressed in ($^\circ$C day$^{-1}$). $A_f$ is a decay coefficient (day$^{-1}$), $K$ is a snow depth reduction coefficient (cm day$^{-1}$), $d_s$ is the depth of the snow cover (expressed as an equivalent water depth (mm)), and $wes$ is the equivalent water depth of a given depth of snow cover. The soil is considered frozen if $F$ is above a critical threshold $F_{crit}$; $F$ is always greater than or equal to 0.

### 2.3 Interception

Interception of rainfall is often simulated using some variation on the classic Rutter model (e.g. Rutter et al. 1971, Gash 1979). Since it is very difficult to obtain reliable estimates of interception-related vegetation characteristics at the continental scale, LISFLOOD follows the even simpler approach of Aston (1979) and Merriam (1960), which requires only two parameters. Interception is estimated as:

$$Int = S_{max} \left[ 1 - \exp \left( - kR \Delta t / S_{max} \right) \right]$$

where $Int$ (mm) is the interception per time step, $S_{max}$ (mm) is the maximum interception, $R$ is the rainfall intensity (mm day$^{-1}$) and the factor $k$ accounts for the density of the vegetation. $S_{max}$ is calculated using the empirical relation (von Hoyningen-Huene 1981):

$$S_{max} = \begin{cases} 0.935 + 0.498LAI - 0.00575LAI^2 & (LAI > 0.1) \\ 0 & (LAI \leq 0.1) \end{cases}$$

where $LAI$ is the average Leaf Area Index ($m^2 m^{-2}$) of each grid cell. Constant $k$ is given by:

$$k = 0.046LAI$$

The value of $Int$ can never exceed the interception storage capacity, which is defined as the difference between $S_{max}$ and the accumulated amount of water that is stored as interception, $Int_{cum}$. Evaporation of intercepted water, $EW_{int}$, occurs at the potential evaporation rate from an open water surface, $EW0$. The maximum evaporation per time step is proportional to the fraction of vegetated area in each pixel (Supit et al. 1994):

$$EW_{max} = EW0 \cdot \left[ 1 - \exp \left( - \kappa_{gb}LAI \right) \right] \Delta t$$

where $EW0$ is the potential evaporation rate from an open water surface ($mm day^{-1}$), and $EW_{max}$ is in (mm) per time step. Constant $\kappa_{gb}$ is the extinction coefficient for global solar radiation. Since evaporation is limited by the amount of water stored on the leaves, the actual amount of evaporation from the interception store equals (Supit et al. 1994):
where $EW_{\text{int}}$ is the actual evaporation from the interception store in (mm) per time step, and $EW_0$ is the potential evaporation rate from an open water surface (mm day$^{-1}$). It is assumed that on average all water in the interception store ($Int_{\text{cum}}$) will have evaporated or fallen to the soil surface as leaf drainage within one day. Leaf drainage is therefore modelled as a linear reservoir with a time constant of one day:

$$D_{\text{int}} = \frac{1}{T_{\text{int}}} \cdot Int_{\text{cum}} \Delta t$$

where $D_{\text{int}}$ is the amount of leaf drainage per time step (mm) and $T_{\text{int}}$ is a time constant for the interception store (days), which is set to one day.

### 2.4 Treatment of impervious areas

If (part of) a pixel is made up of built-up area this will influence that pixel’s water-balance. The ‘direct runoff fraction’ parameter ($f_{dr}$) defines the fraction of a pixel that is impervious. For impervious areas, it is assumed that:

1. any water that reaches the surface is added directly to surface runoff;
2. the storage capacity of the soil is zero (i.e. no soil moisture storage in the direct runoff fraction);
3. there is no groundwater storage.

The same assumptions are made for open water bodies (e.g. lakes), which are included in $f_{dr}$. Unless stated otherwise, the description of all subsurface processes below (evaporation, transpiration, infiltration, preferential flow, soil moisture redistribution and groundwater flow) are valid for the pervious domain of each pixel (i.e. $1 - f_{dr}$) only. In the pervious fraction of each pixel ($1 - f_{dr}$), the amount of water that is available for infiltration, $W_{av}$ (mm) equals:

$$W_{av} = R \Delta t + M + D_{\text{int}} - Int$$

where $R$ is the rainfall intensity (mm day$^{-1}$), and $M$, $D_{\text{int}}$ and $Int$ are the amounts of snowmelt, leaf drainage and interception, respectively (all in (mm) per time step). Since no infiltration can take place in each pixel’s ‘direct runoff fraction’, direct runoff is calculated as:

$$R_d = f_{dr} \cdot W_{av}$$

with $R_d$ is in (mm) per time step. Note here that $W_{av}$ is valid for the pervious fraction only, whereas $R_d$ is valid for the direct runoff fraction.

### 2.5 Evapotranspiration

Water uptake and transpiration by vegetation and direct evaporation from the soil surface are modelled as two separate processes. Our approach is largely based on Supit et al. (1994) and Supit and van der Goot (2003), which is in turn an adaptation of the widely used FAO Penman-Monteith method (Allen et al. 1998). The main reason for using this method is that it is a widely accepted approach. Moreover, it uses meteorological forcing variables that are identical to the ones stored in an
existing pan-European agro-meteorological database (Rijks et al. 1998). This means that the historical data in the database can be used directly as input to LISFLOOD. The maximum transpiration per time step (mm) is calculated as:

\[ T_{\text{max}} = k_{\text{crop}} \cdot ET_0 \cdot [1 - \exp(-\kappa_{gb} \cdot LAI)] \Delta t - EW_{\text{int}} \] (11)

where \( ET_0 \) is the potential (reference) evapotranspiration rate (mm day\(^{-1}\)), and \( k_{\text{crop}} \) is a crop coefficient. Note that the energy that has been ‘consumed’ already for the evaporation of intercepted water is simply subtracted here in order to respect the overall energy balance. The actual transpiration rate is reduced in case of water stress. A reduction factor is applied to account for this:

\[ r_{WS} = \frac{w_1 - w_{wp1}}{w_{crit1} - w_{wp1}} \] (12)

where \( w_1 \) is the amount of moisture in the upper soil layer (mm), \( w_{wp1} \) (mm) is the amount of soil moisture at wilting point (pF 4.2) and \( w_{crit1} \) (mm) is the amount of moisture below which water uptake is reduced and plants start closing their stomata.

The critical amount of soil moisture is calculated as:

\[ w_{crit1} = (1 - p) \cdot (w_{fc1} - w_{wp1}) + w_{wp1} \] (13)

where \( w_{fc1} \) (mm) is the amount of soil moisture at field capacity and \( p \) is the soil water depletion fraction. Parameter \( p \) represents the fraction of soil moisture between \( w_{fc1} \) and \( w_{wp1} \) that can be extracted from the soil without reducing the transpiration rate, and its value is a function of both \( ET_0 \) and land cover (details on this can be found in Supit and Van Der Goot (2003)). Reduction factor \( r_{WS} \) is allowed to assume values between 0 and 1 only. The actual transpiration \( T_a \) is now calculated as:

\[ T_a = r_{WS} \cdot T_{\text{max}} \] (14)

with both \( T_a \) and \( T_{\text{max}} \) in (mm).

The maximum amount of evaporation from the soil surface equals the maximum evaporation from a shaded soil surface, \( ES_{\text{max}} \) (mm), which is computed as:

\[ ES_{\text{max}} = ES_0 \exp(-\kappa_{gb} \cdot LAI) \Delta t \] (15)

where \( ES_0 \) is the potential evaporation rate from bare soil surface (mm day\(^{-1}\)). The actual evaporation from the soil mainly depends on the amount of soil moisture near the soil surface: evaporation decreases as the topsoil dries out. This is simulated using a reduction factor which is a function of the number of days since the last rain storm (Stroosnijder 1982, 1987):

\[ ES_a = ES_{\text{max}} \left( \sqrt{D_{slr}} - \sqrt{D_{slr} - 1} \right) \] (16)

Here variable \( D_{slr} \) represents the number of days since the last rain event. Its value accumulates over time: if the amount of water that is available for infiltration \( (W_{av}) \) remains below a critical threshold \( (W_{crit}) \), it increases by an amount of \( \Delta t \) (days) for each time step. It is reset to 1 only if the critical amount of water is exceeded. The actual soil evaporation is always the smallest value out of the result of the equation above and the available amount of moisture in the soil, i.e.:

\[ ES_a = \min(ES_a, w_1 - w_{r1}) \] (17)
where $w_1$ (mm) is the amount of moisture in the upper soil layer and $w_{r1}$ (mm) is the residual amount of soil moisture.

### 2.6 Infiltration, preferential flow and surface runoff

The infiltration capacity of the soil is estimated using the widely-used Xinanjiang (also known as VIC/ARNO) method (e.g. Zhao and Liu 1995, Todini 1996). In contrast to most other infiltration models, it explicitly takes into account sub-pixel heterogeneity of infiltration capacity, which is essential for large-scale runoff modelling. It does so by assuming that the fraction of a grid cell that is contributing to surface runoff is related to the total amount of soil moisture, and that this relationship can be described through a non-linear distribution function. For any grid cell, if $w_1$ is the total moisture storage in the upper soil layer and $w_{s1}$ is the maximum storage, the corresponding saturated fraction $A_s$ is approximated by the following distribution function:

$$A_s = 1 - \left(1 - \frac{w_1}{w_{s1}}\right)^b$$  \quad (18)

where $b$ is a dimensionless empirical shape parameter, which is typically used as a calibration constant. Note that $A_s$ is expressed as a fraction of the pervious fraction only. The infiltration capacity $INF_{pot}$ (mm) is a function of $w_{s1}$ and $A_s$:

$$INF_{pot} = \frac{w_{s1}}{b+1} - \frac{w_{s1}}{b+1} \left[1 - (1 - A_s)^{b+1}\right]$$  \quad (19)

Note that the shape parameter $b$ is related to the heterogeneity within each grid cell. For a totally homogeneous grid cell $b$ approaches zero, which reduces the above equations to a simple ‘overflowing bucket’ model. For the simulation of preferential flow – i.e. flow that bypasses the soil matrix and drains directly to the groundwater – no generally accepted equations exist. Because ignoring preferential flow completely will lead to unrealistic model behaviour during extreme rainfall conditions, we adopted the following simple approach. During each time step, a fraction of the water that is available for infiltration is added to the groundwater directly, thereby bypassing the soil matrix. It is assumed that this fraction is a power function of the relative saturation of the topsoil. This yields an equation that is somewhat similar to the excess soil water equation used in the HBV model (e.g. Lindström et al. 1997):

$$D_{pref, gw} = W_{av} \left(\frac{w_1}{w_{s1}}\right)^{c_{pref}}$$  \quad (20)

where $D_{pref, gw}$ is the amount of preferential flow per time step (mm), $W_{av}$ is the amount of water that is available for infiltration, and $c_{pref}$ is an empirical shape parameter, which is used as a calibration constant. The equation results in a preferential flow component that becomes increasingly important as the soil gets wetter. The actual infiltration $INF_{act}$ (mm) per time step is now calculated as:

$$INF_{act} = \min(INF_{pot}, W_{av} - D_{pref, gw})$$  \quad (21)

Finally, the surface runoff $R_s$ (mm) is calculated as:

$$R_s = R_d + (1 - f_{dr}) \cdot (W_{av} - D_{pref, gw} - INF_{act})$$  \quad (22)
where $R_d$ is the direct runoff (generated in the pixel’s ‘direct runoff fraction’). Equation (22) thus gives the surface runoff for the whole pixel ( pervious + impervious fraction).

### 2.7 Soil moisture flow

Soil moisture fluxes in the unsaturated zone are often simulated using Darcy’s law for one-dimensional vertical flow. The flux ($D$) out of a soil layer (e.g. upper soil layer, lower soil layer) is then given by:

$$D = -K(\theta) \left[ \frac{\partial h(\theta)}{\partial z} - 1 \right]$$

where $D$ is in (mm day$^{-1}$), $K(\theta)$ is the hydraulic conductivity (mm day$^{-1}$), $\theta$ is the soil’s volumetric moisture content (mm$^3$ mm$^{-3}$) and $\partial h(\theta)/\partial z$ is the matric potential gradient. Equation (23) describes a flux that can either be in downward (positive) or upward (negative) direction. In the latter case it describes capillary rise. However, the solution of the equation is numerically complex and computationally very demanding. Because of this, we make the simplifying assumption that the movement of moisture through the soil is entirely gravity-driven. If we assume a matric potential gradient of zero, equation (23) describes a flow that is always in downward direction, at a rate that equals the conductivity of the soil. The relationship between hydraulic conductivity and soil moisture status can be described by the van Genuchten equation (van Genuchten 1980), here re-written in terms of mm water slice:

$$D = K(w) = K_s \left( \frac{w - w_r}{w_s - w_r} \right)^{1/m} \left[ 1 - \left( \frac{w - w_r}{w_s - w_r} \right)^{1/m} \right]^m $$(24)

Here, $K_s$ is the saturated conductivity of the soil (mm day$^{-1}$); $w$, $w_r$ and $w_s$ are the actual, residual and maximum amounts of moisture in the soil respectively (all in (mm)). Parameter $m$ is calculated from the pore-size index, $\lambda$, which is related to soil texture:

$$m = \frac{\lambda}{\lambda + 1}$$

Equation (24) is used to calculate the fluxes from the upper to the lower soil layer ($D_{1,2}$), and from the lower soil layer to the groundwater system ($D_{2, gw}$), respectively. Because both fluxes are always in downward direction, capillary rise is not simulated. For large values of $At$, the equation can produce soil moisture fluxes that exceed the available soil moisture. Therefore, the equation is solved on a smaller time interval, the size of which is determined by a Courant-type numerical stability criterion. The routine is computationally quite efficient: running the model on a daily time step, the number of iterations needed rarely exceeds 9, and is usually 1 or 2.

### 2.8 Subsurface flow

Subsurface storage and transport are modelled using two parallel linear reservoirs, which is similar to the approach used in the HBV-96 model (Lindström et al. 1997), and many other rainfall-runoff models. The upper zone represents a quick runoff
component, which includes fast groundwater and subsurface flow through macro-pores in the soil. The lower zone represents the slow groundwater component that generates the base flow. The outflow from the upper zone to the channel, $Q_{uz}$ (mm) equals:

$$Q_{uz} = \frac{1}{T_{uz}} \cdot UZ \Delta t$$  \hspace{1cm} (26)$$

where $T_{uz}$ is a reservoir constant (days), and $UZ$ is the amount of water that is stored in the upper zone (mm). Likewise, the outflow from the lower zone is given by:

$$Q_{lz} = \frac{1}{T_{lz}} \cdot LZ \Delta t$$  \hspace{1cm} (27)$$

Here, $T_{lz}$ is again a reservoir constant (days), and $LZ$ is the amount of water that is stored in the lower zone (mm). The values of both $T_{uz}$ and $T_{lz}$ are obtained by calibration. The upper zone also provides the inflow into the lower zone. For each time step, a fixed amount of water percolates from the upper to the lower zone:

$$D_{uz, lz} = \min \left( GW_{perc} \Delta t, UZ \right)$$ \hspace{1cm} (28)$$

where $GW_{perc}$ (mm day$^{-1}$) is a user-defined value that can be used as a calibration constant. It is usually not unrealistic to treat the lower groundwater zone as a system with a closed lower boundary (i.e. water is either stored, or added to the channel). For situations in which this is not the case, it is possible to treat a fixed fraction of $Q_{lz}$ as a loss, $D_{loss}$ (mm), out of the lower zone:

$$D_{loss} = f_{loss} \cdot Q_{lz}$$  \hspace{1cm} (29)$$

The loss fraction, $f_{loss}$, equals 0 for a completely closed lower boundary. If $f_{loss}$ is set to 1, all outflow from the lower zone is treated as a loss. Physically, the loss term could represent water that is either lost to deep groundwater systems (that do not necessarily follow catchment boundaries), or groundwater extraction wells. At each time step, the amounts of water in the upper and lower zone are updated for the in- and outgoing fluxes, i.e.:

$$UZ_t = UZ_{t-1} + D_{2, gw} - D_{uz, lz} - Q_{uz}$$ \hspace{1cm} (30)$$

$$LZ_t = LZ_{t-1} + D_{uz, lz} - Q_{lz}$$  \hspace{1cm} (31)$$

The slow overall response of the lower zone implies that it is prone to initialisation problems that may lead to artificial trends in the simulated baseflow. The model has a special option to calculate the lower zone’s steady-state storage (which is a function of the model parameters and the meteorological forcing). Starting a simulation with this steady-state level guarantees the absence of any such initialisation issues.

2.9 Hillslope and channel routing

Routing is done in two stages. First, the generated runoff in each pixel is routed to the nearest downstream channel pixel. Surface runoff is routed using a four-point
implicit finite-difference solution of the kinematic wave equations (Chow et al. 1988). As for the sub-surface runoff, all water that flows out of the upper- and lower groundwater zones ($Q_{uz}$ and $(Q_{lz} - D_{loss})$) is routed to the nearest downstream channel pixel within one time step. This effectively means that, as far as sub-surface runoff is concerned, we treat the upstream ‘land surface’ pixels of each river pixel as spatially lumped units. This lumping does not influence the simulation of streamflow in the river channel much, provided that the flow paths through the contributing surface areas are not too long. For our existing input data sets at 1 km resolution, the length of these flow paths rarely exceeds 10 km, and is usually much less. Thus, for the simulation of large river basins this approach seems reasonable. (At higher spatial resolutions one would typically use a denser river network as well, which would in turn ‘push back’ the effect of the lumping to another more upstream level.) Finally, the water in each channel pixel is routed through the channel network. By default we again employ the four-point implicit finite-difference solution of the kinematic wave equations. LISFLOOD is capable of full dynamic wave routing as well, although our implementation of the dynamic wave equations requires detailed channel cross-section data. Since these data are not readily available for most rivers, the dynamic wave is included as an option. A number of additional options exist to model special structures within the channel network. First of all, large lakes that are part of the channel network can be simulated as points in the channel network. Lake inflow equals the simulated discharge upstream of the lake, and a rating curve is used to compute the lake outflow into the downstream channel reach (see e.g. Maidment 1993):

$$Q_{lake} = A(H - H_0)^B$$

(32)

where $Q_{lake}$ is the lake outflow ($\text{m}^3\text{s}^{-1}$), $H$ is the water level in the lake (m), $H_0$ is the water level for which the lake outflow is zero (m), and $A$ and $B$ are empirical constants. Lake evaporation is simulated at the potential rate of an open water surface, $EW0$ (mm day$^{-1}$). The effect of using the lake routine is an attenuation of the routed discharge wave. A separate option exists for the simulation of regulated reservoirs. Reservoir outflow is calculated from user-specified rules that define reservoir behaviour as a function of filling level and upstream inflow. Finally, it is possible to feed (measured) inflow hydrographs directly into the channel at selected locations, which is useful in cases where one only wants to simulate the downstream part of a river basin. For more details on these options we refer to van der Knijff and de Roo (2008).

3. Model parameterisation

Table 1 lists all parameters that are needed by LISFLOOD. For the majority of these parameters, reasonable prior estimates can be made. For example, most soil and land-use related parameters can be estimated from existing data sets such as the Soil Geographical Database of Europe (King et al. 1997), the HYPRES database on hydraulic soil properties (Wöstén et al. 1999), and the CORINE land use database (CEC 1993). An important parameter is Leaf Area Index ($LAI$). Several techniques exist to estimate spatiotemporal variations in $LAI$ from spaceborne satellite imagery (de Jong and Jetten 2007). Besides this, data sets such as the MODIS-LAI product (Myneni et al. 2002) provide readily available global coverage of $LAI$. Since LISFLOOD takes its $LAI$ input as a stack of spatial grids, with each grid defined at user-defined time steps, such remote sensing-derived $LAI$ products can be used
directly in the model. Other parameters, such as those that are related to snowmelt and frost, can be estimated from published literature values. The remaining parameters need to be estimated by calibrating the model against observed discharge records. As an example, a study by Feyen et al. (2007) addressed the calibration of LISFLOOD for the Dutch–Belgian–French Meuse catchment. For this study, all but five parameters were estimated from prior data. The remaining parameters \((T_{uz}, b, c_{pref}, GW_{perc} \text{ and } T_{lz})\) were estimated by calibration against observed discharge, using the Shuffled Complex Evolution Metropolis global optimization algorithm. The resulting posterior parameter distributions were used to assess the sensitivity of LISFLOOD to the calibration parameters, and to construct uncertainty intervals.

### Table 1. LISFLOOD model parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_{snow})</td>
<td>Snow correction factor</td>
<td>—</td>
</tr>
<tr>
<td>(T_{snow})</td>
<td>Temperature below which precipitation is treated as snow</td>
<td>°C</td>
</tr>
<tr>
<td>(T_{m})</td>
<td>Temperature above which snowmelt occurs</td>
<td>°C</td>
</tr>
<tr>
<td>(C_m)</td>
<td>Snowmelt degree-day factor</td>
<td>mm °C⁻¹ day⁻¹</td>
</tr>
<tr>
<td>(L)</td>
<td>Temperature lapse rate</td>
<td>°C m⁻¹</td>
</tr>
<tr>
<td>(A_f)</td>
<td>Frost index decay coefficient</td>
<td>day⁻¹</td>
</tr>
<tr>
<td>(K)</td>
<td>Frost index snow depth reduction coefficient</td>
<td>cm⁻¹</td>
</tr>
<tr>
<td>(w_{es})</td>
<td>Snow water equivalent</td>
<td>—</td>
</tr>
<tr>
<td>(F_{crit})</td>
<td>Value of frost index above which soil is considered frozen</td>
<td>°C day⁻¹</td>
</tr>
<tr>
<td>(f_{dr})</td>
<td>Direct runoff fraction</td>
<td>—</td>
</tr>
<tr>
<td>(LAI)</td>
<td>Leaf Area Index (as a function of time)</td>
<td>m² m⁻²</td>
</tr>
<tr>
<td>(\kappa_{gb})</td>
<td>Extinction coefficient for global solar radiation</td>
<td>—</td>
</tr>
<tr>
<td>(k_{crop})</td>
<td>Crop coefficient</td>
<td>—</td>
</tr>
<tr>
<td>(T_i)</td>
<td>Time constant of rainfall interception store</td>
<td>days</td>
</tr>
<tr>
<td>(W_{crit})</td>
<td>Threshold for resetting DSLR in soil evaporation reduction equation</td>
<td>mm</td>
</tr>
<tr>
<td>(n_s)</td>
<td>Surface Manning’s roughness coefficient</td>
<td>—</td>
</tr>
<tr>
<td>(d_r)</td>
<td>Rooting depth</td>
<td>cm</td>
</tr>
<tr>
<td>(b)</td>
<td>Infiltration constant</td>
<td>—</td>
</tr>
<tr>
<td>(c_{pref})</td>
<td>Preferential flow constant</td>
<td>—</td>
</tr>
<tr>
<td>(K_{1,2})</td>
<td>Saturated hydraulic conductivity layer 1, 2</td>
<td>cm day⁻¹</td>
</tr>
<tr>
<td>(\theta_{1,2})</td>
<td>Saturated volumetric moisture content layer 1, 2</td>
<td>mm³ mm⁻³</td>
</tr>
<tr>
<td>(\theta_{1,2})</td>
<td>Residual volumetric moisture content layer 1, 2</td>
<td>mm³ mm⁻³</td>
</tr>
<tr>
<td>(\lambda_{1,2})</td>
<td>Pore-size index layer 1, 2</td>
<td>—</td>
</tr>
<tr>
<td>(\alpha_{1,2})</td>
<td>Constant in soil water retention equation layer 1, 2</td>
<td>—</td>
</tr>
<tr>
<td>(d_s)</td>
<td>Soil depth</td>
<td>cm</td>
</tr>
<tr>
<td>(T_{uz})</td>
<td>Upper zone time constant</td>
<td>days</td>
</tr>
<tr>
<td>(T_{lz})</td>
<td>Lower zone time constant</td>
<td>days</td>
</tr>
<tr>
<td>(GW_{perc})</td>
<td>Maximum rate of percolation from upper to lower zone</td>
<td>mm day⁻¹</td>
</tr>
<tr>
<td>(f_{loss})</td>
<td>Groundwater loss fraction</td>
<td>—</td>
</tr>
<tr>
<td>(g_{ch})</td>
<td>Channel bed gradient</td>
<td>m m⁻¹</td>
</tr>
<tr>
<td>(n_{ch})</td>
<td>Channel Manning’s roughness coefficient</td>
<td>—</td>
</tr>
<tr>
<td>(l_{ch})</td>
<td>Length of channel element</td>
<td>m</td>
</tr>
<tr>
<td>(w_{ch})</td>
<td>Channel bottom width</td>
<td>m</td>
</tr>
<tr>
<td>(s_{ch})</td>
<td>Channel side-slope</td>
<td>m m⁻¹</td>
</tr>
<tr>
<td>(bf_{ch})</td>
<td>Channel bankfull depth</td>
<td>m</td>
</tr>
</tbody>
</table>
around the simulated hydrographs. The analysis showed that the model is particularly sensitive to the parameters that are involved in the generation of fast runoff: $T_{uc}$ and $c_{pref}$, and to a lesser extent also $b$. Less sensitivity was found for $GW_{perc}$ and $T_{lz}$, which both control slow runoff mechanisms.

In the absence of prior data of sufficient quality it may be necessary to include other parameters in the calibration procedure. This is no problem, and the model poses no restrictions on its users as to which parameters are used for calibration and which ones are ‘fixed’ using prior data. Also, each parameter can be defined either as a single value, or as a spatially distributed grid. Thus it is possible to account for within-basin variability, and multiple river basins may even be combined in one single model run. Another study by Feyen et al. (2008) demonstrates both principles, using a semi-distributed parameterisation scheme for the calibration of the Czech–Austrian–Slovak Morava basin. Apart from taking into account the spatial variation of the calibration parameters, they also included two additional snow-related parameters in their calibration exercise. The main danger of LISFLOOD’s flexibility regarding the selection of its calibration parameters is an increased risk of over-parameterisation problems. However, our experiences with the model so far have shown that it is very difficult – if not impossible – to define any fixed set of calibration parameters that ‘work’ in all possible cases. This can be largely explained by the wide range of climatic and hydrological regimes that can be found across Europe. For instance, for most catchments in southern Europe LISFLOOD’s snow and frost related parameters are completely irrelevant, whereas they may control the dominant hydrological processes in Scandinavia. Nevertheless, over-parameterisation is a real risk, and it is a good practice to limit the dimensionality of the calibration exercise by using prior data whenever possible.

4. Technical implementation

LISFLOOD is written in a combination of the PCRaster Dynamic Modelling Language (Wesseling et al. 1996, Karssenberg 2002) and the Python scripting language. PCRaster is a raster geographical information system (GIS) that has its own embedded dynamic programming language. Karssenberg (2002) gives an in-depth discussion of the advantages of high-level languages such as PCRaster for the development of distributed hydrological models. It is beyond the scope of this paper to repeat all his arguments here. In short, in contrast to low-level languages such as C++ or FORTRAN, PCRaster hides any low-level operations such as file in- and output and memory management from the programmer, and provides a level of abstraction that is more at par with the level of thinking of hydrologists. This results in code that is shorter, easier to read, maintain, modify and re-use. Since all such operations are handled by generic, highly optimized built-in functions of PCRaster, this also results in code that is very stable. When a relatively complex model such as ours is applied to very large datasets this last issue becomes particularly important, especially if the model is used as part of an operational system (the European Flood Alert System (EFAS) mentioned in section 5 is a good example of this). Finally, since PCRaster comes with a host of visualisation tools, these are readily available to display and analyse LISFLOOD’s output.

In spite of these benefits, until recently some limitations of the software inhibited the development of a fully operational simulation model in ‘pure’ PCRaster. Most importantly, an operational setting requires that model users (who may not know anything about the inner workings of the model) can exert some control over the
model flow. For instance, the EFAS (discussed in section 5) is based entirely on the analysis of grids of simulated discharge. Most model users will only need discharge time series at selected gauge locations. Other model users may also be interested in intermediate state variables, such as soil moisture, and want these as either grids, time series at points, or spatially averaged values over the area draining to each gauge location. Reporting all these variables (in all possible formats) would greatly increase the computation time, as well as the amount of disk space used. Since PCRaster does not have any mechanisms to let the user activate or disable parts of the model (such as file write statements) without actually getting into the source code, this poses some restrictions in an operational setting. We eliminated these restrictions by writing a simple software application that acts as a wrapper around PCRaster. The wrapper is written in the Python scripting language (Python 2008). The basic idea is shown in figure 2. For each LISFLOOD run, the wrapper performs the following sequence of tasks:

1. **Read and analyse the LISFLOOD ‘master code’**, which contains the source code of all process descriptions. The master code is similar to a conventional PCRaster script (see e.g. Wesseling *et al.*, 1996). However, the main difference is that it uses a special xml structure that allows a high degree of modularisation of the code. Although we will not provide a detailed description of this structure here, one particular feature is that all blocks of code that make up the optional model components (e.g. the reporting of discharge grids) are defined as separate xml elements. Each ‘option’ element has attributes that define under which condition it should be ‘switched on’, and whether it should be ‘switched on’ by default (or not).

2. **Read and analyse a settings file**. The settings file contains all the information that is needed for a model run, such as the names and locations of all in- and output files, the simulation time step, and parameter values. In addition to this, users can add switches to activate optional model components.

3. **Generate a PCRaster script** that contains all required options and user settings (as defined in the settings file).

4. **Launch the generated script** using PCRaster’s computational engine ‘pcrcalc’.

5. **Delete the script** once the model has finished.

![Figure 2](image-url)  
**Figure 2.** Overview of the Python wrapper for LISFLOOD.
One noteworthy feature is that the format of the ‘master code’ is completely generic. This means that updates or other modifications to the model can be done by changing the ‘master code’, without any need whatsoever to change the wrapper routine. Because of this, both maintenance and further development of the code are just as straightforward as they would be in a ‘native’ PCRaster script.

In order to give a rough indication of LISFLOOD’s computational performance, we prepared a four-year, daily time-step (1430 steps) water balance simulation for the full Elbe river basin. At 1-km resolution, this resulted in a simulation grid of 140,309 cells (we will discuss the study area in greater detail in section 6.1). We ran the simulation under Windows XP Professional on a standard desktop PC with a 2.40 GHz Intel Xeon processor and 1.5 GB internal memory. The total time needed for this model run was 3 hours and 18 minutes. Computing times of this order of magnitude may make the use of automatic calibration tools seem somewhat prohibitive, since such routines typically require hundreds of model runs. However, because the calibration of LISFLOOD is usually done in a spatially distributed fashion, large basins such as this one are usually split up in smaller sub-catchments, each of which is then calibrated separately. The resulting procedure lends itself perfectly to the use of parallel computing clusters, and in fact these have been used extensively for most recent calibration work (Feyen et al. 2007, 2008). Currently the model runs under 32-bit Windows and under a number of Linux distributions. Ports to other operating systems may follow in the near future.

5. Data management

In this section we will describe what types of data are used with LISFLOOD. We also explain how the model’s in- and output data can be exchanged with other software applications.

5.1 Map data

Most input to LISFLOOD is defined in the form of PCRaster maps. Complete sets of LISFLOOD base maps that cover the whole of Europe have been created at both 1 and 5-km grid resolution. For any given European catchment, a ready-to-use setup can be created by simply extracting the base maps for the area of interest. This can be done using PCRaster’s standard data management tools. In addition to this a set of Python scripts has been written around these tools, which completely automates the map extraction process. It is also possible to create new LISFLOOD base maps from scratch, or to edit existing maps. This can be done using any conventional rater GIS package, such as ArcGIS or GRASS. PCRaster includes tools for importing and exporting map data from and to a number of ASCII formats, including ESRI’s popular ASCIIGRID format. In addition, the PCRaster map format is supported by the Geospatial Data Abstraction Library (GDAL) library (GDAL 2008). This is an open-source translator library for geospatial raster data, which can be used for conversions between many different raster formats.

Since all of LISFLOOD’s state and rate variables can be written to maps as well, the same tools can be used to export the model’s output to other software applications. A set of custom-made tools has been written for pre-processing and managing the model’s meteorological input data, which are all fed into the model as stacks of PCRaster maps. First of all, depending on the data source, raw meteorological data are either provided as point observations or as interpolated grids. For the former
case, a set of tools and PCRaster scripts is available for the spatial interpolation of such point data. In the latter case, the aforementioned importer tools can be used. Second, most meteorological data sets do not provide direct estimates of the potential evapo(transpi)ration rates $ET_0$, $ES_0$ and $EW_0$. However, these variables can be derived from the surface radiation budget, which can be calculated from standard meteorological observations. To this end, a separate pre-processing application (LISVAP) has been developed, which can be used in conjunction with LISFLOOD. A detailed description of the LISVAP software can be found in van der Knijff (2008).

5.2 Table and time series data

Model parameters that are directly linked to soil surface texture and land use classes are defined in lookup tables, which are plain text files that can be viewed (and edited, if necessary) in any text editor. All of LISFLOOD’s state and rate variables can be written to time series as well. Time series are written as plain text files, which can be easily imported in e.g. spreadsheet software. They can be reported in two different ways. First, variables can be reported at user-defined locations on a map. These locations can be either points or areas. In the latter case, the time series file contains areal averages. Second, at each discharge gauge location, spatial averages can be calculated. In this case, each variable is averaged over the upstream contributing area of each gauge. This is particularly useful for getting a summary view of all components of the water balance at each gauge location.

6. Applications

In this section we will first review some applications of LISFLOOD that have appeared in the literature. We also present a brief case study for the Elbe catchment. The model is at the core of the EFAS, which is described in detail by Thielen et al. (2008) and Ramos et al. (2007). EFAS uses both deterministic and probabilistic weather forecasts, which are used as input to LISFLOOD. For each forecast, simulated discharges are evaluated in terms of exceedance of predefined flood alert thresholds, which are in turn based on a statistical analysis of long-term time series of simulated discharge. The system was set up to provide early flood warnings in European transnational river basins, and has been in pre-operational testing mode since 2005. The whole of Europe is included, and all major European river basins are combined in one single model setup. Each basin was calibrated using an automatic algorithm that combines an adaptive partition-based search and a downhill simplex method (Szabó 2006). A detailed case study of an actual flood event in the Czech part of the Elbe river basin that was predicted by EFAS can be found in Younis et al. (2008b). A separate study by Younis et al. (2008a) focuses on the prediction of flash floods in southern France.

Some recent studies have used the model to evaluate river discharge under a changing climate. Gouweleeuw et al. (2004) used atmospheric output data from the rerun of the ECMWF Global Circulation model (ERA40) to force a model run over the period 1958–2002 for the whole of Europe. This allowed them to generate an extensive pan-European database of time series of historic river flow. Dankers et al. (2007) used the climatic output of another, high resolution climate model to simulate river discharge in the Upper Danube basin in central Europe. Besides a comparison of simulated discharge for different climatic input resolutions, they also
evaluated various future climate change scenarios. By doing so, they were able to make some tentative predictions of the impact of future climate changes on the occurrence of floods. As an example, figure 3 shows the results of a simulation for the Danube at Bratislava. Note that in this case the model was calibrated using observed discharge data from a different time period (October 1994–September 1997). In terms of long-term water balance, the model shows a good agreement with the observed discharge. However, zooming in at the August 2002 flood reveals that in this case the model overestimates the flood peak, and gives a response that is too fast. These discrepancies can be explained by the fact that the main Danube upstream of Bratislava is heavily regulated. A series of locks, reservoirs and artificial channels effectively allowed the water authorities to reduce and delay the flood peak, and none of these structures were accounted for in this simulation.

6.1 Elbe case study

The Elbe ranks as the fourth largest river of western and central Europe. The main river has a total length of about 1100 km, draining a basin area of about 148,000 km². The basin comprises parts of Poland, Austria, the Czech Republic and Germany, although the main river and 99% of its drainage area are confined to the Czech Republic and Germany. We created a set of input data for the whole basin at 1-km grid resolution. High-resolution meteorological data were provided by the Czech Meteorological Institute and the German federal meteorological authorities, and we used these data to generate two sets of interpolated meteorological input grids: one for the period 1994–1998, and a second one for the period 1999–2002. We used observed discharge data at 20 gauge locations to calibrate the model for the years 1994–1998, with the first year being used as a warm-up period. For calibration we used an automatic algorithm that combines an adaptive partition-based search and a downhill simplex method (Szabó 2006). We applied the algorithm in a semi-distributed fashion, using spatial

Figure 3. LISFLOOD simulation of the Danube river basin. Figure shows simulated and observed discharge at Bratislava for the validation period October 1998–September 2002 (inset shows a more detailed view for the year 2002). Figure redrawn after Dankers et al. (2007).
units that are defined by the sub-basins draining to each gauge location. We used the 1999–2002 period – which includes the August 2002 flood – for validation, again using the first year for warm-up. As an example, figure 5 shows the results of both calibration and validation for the Dresden gauge, which is representative of most of the central Elbe section. As for the validation run, it is noteworthy to point out that the peak of the August 2002 flood is approximated rather well by the model, although the timing of the peak is represented less accurately. Table 2 summarizes the results for the whole basin. The ‘goodness of fit’ at each gauging station is characterized using the following performance statistics: (i) root mean square error:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (O_i - P_i)^2}$$  \hspace{1cm} (33)

and (ii) Nash and Sutcliffe efficiency (Nash and Sutcliffe 1975):

$$E = 1 - \frac{\sum_{i=1}^{N} (O_i - P_i)^2}{\sum_{i=1}^{N} (O_i - \bar{O}_i)^2}$$  \hspace{1cm} (34)

In both indices $O_i$ denotes the observed discharge at time step $i$; $\bar{O}_i$ is the mean observed discharge; $P_i$ is the simulated discharge at time step $i$, and $N$ is the number of time steps. A value of $E=1$ indicates a perfect agreement between observed and simulated discharge, and for $E<0$ the mean of the observations is a better predictor than the simulated values (Legates and McCabe 1999). From these results we can make a number of observations. First of all, for many gauging stations the statistics indicate that the model performs better over the validation period than it does over the calibration period. One would usually expect the opposite to be true. However, looking again at figure 5 we can seen that the validation period can be characterized as very dry, with a wet period around the start of 2002, followed by another relatively dry interval which is in turn followed by the Elbe flood of August 2002. As a result, most of the variance in this time series originates from the differences between the long baseflow-dominated period and the two high-discharge intervals. In contrast to this, the calibration period has much more short-term variation. In this view, the ‘better’ values of the performance statistics are hardly surprising, because the temporal discharge pattern during the validation period is simply less challenging to reproduce with a model such as LISFLOOD. Also, data from more meteorological measurement stations were available for the validation period than for the calibration period, and this may have contributed to the better results for the validation as well. As a second observation, we can see a reduced model performance for the most downstream stretch of the main Elbe. This is mainly because this part of the river is heavily regulated, and the presence of artificial structures such as dams and reservoirs was not taken into account in any of our simulations. In spite of this, for most gauging stations the model results show a reasonable – and often good – agreement with respect to the observed discharge.

7. Conclusions

In this paper we presented the LISFLOOD model. We described its general characteristics, and discussed how the various hydrological processes at the hillslope
and channel level are simulated. We gave an overview of the parameters that are needed, and explained how these parameters can be estimated. We also discussed how we implemented the model using a combination of the PCRaster Dynamic Modelling Language and Python, and we described the management of in- and output data. We provided an overview of published case studies that have employed the model, and presented a new case study for the Elbe basin to illustrate some of the possible uses of the model. These examples demonstrate the potential of LISFLOOD for a variety of application fields, including operational flood forecasting, climate scenario studies, and the simulation of historic river discharge. They also show some of the current limitations, most importantly a reduced model performance in (mostly far downstream) channel reaches that are heavily regulated. Although LISFLOOD includes optional modules for the simulation of lakes and reservoirs, the simulation of man-made structures remains a difficult issue. This is
Figure 5. Observed and simulated discharge at Dresden for calibration (top) and validation (bottom) period. Note that both model runs were preceded by a one-year warm-up period, which is not shown here. Inset bottom graph zooms in on the August 2002 flood.

Table 2. Summary performance statistics Elbe calibration and validation. (RMSE=Root Mean Squared Error)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E$</td>
<td>RMSE, m³/s</td>
<td>$E$</td>
<td>RMSE, m³/s</td>
</tr>
<tr>
<td>Ceske Budejovice (Vltava)</td>
<td>0.44</td>
<td>15.26</td>
<td>0.85</td>
<td>26.56</td>
</tr>
<tr>
<td>Bechyne (Luznice)</td>
<td>0.70</td>
<td>11.34</td>
<td>0.88</td>
<td>16.96</td>
</tr>
<tr>
<td>Pisek (Otava)</td>
<td>0.53</td>
<td>11.88</td>
<td>0.86</td>
<td>21.30</td>
</tr>
<tr>
<td>Porice (Sazava)</td>
<td>0.64</td>
<td>9.65</td>
<td>0.44</td>
<td>20.47</td>
</tr>
<tr>
<td>Beroun (Berounka)</td>
<td>0.64</td>
<td>16.94</td>
<td>0.83</td>
<td>39.15</td>
</tr>
<tr>
<td>Praha M. Chuchle (Vltava)</td>
<td>0.65</td>
<td>54.68</td>
<td>0.94</td>
<td>75.15</td>
</tr>
<tr>
<td>Brandys (Elbe)</td>
<td>0.75</td>
<td>40.13</td>
<td>0.81</td>
<td>46.87</td>
</tr>
<tr>
<td>Louny (Ohre)</td>
<td>0.56</td>
<td>11.89</td>
<td>0.70</td>
<td>20.92</td>
</tr>
<tr>
<td>Usti nad Labem (Elbe)</td>
<td>0.72</td>
<td>90.51</td>
<td>0.81</td>
<td>163.57</td>
</tr>
<tr>
<td>Dresden (Elbe)</td>
<td>0.73</td>
<td>98.06</td>
<td>0.74</td>
<td>201.03</td>
</tr>
<tr>
<td>Torgau (Elbe)</td>
<td>0.75</td>
<td>100.94</td>
<td>0.70</td>
<td>221.02</td>
</tr>
<tr>
<td>Wittenberg (Elbe)</td>
<td>0.71</td>
<td>95.63</td>
<td>0.64</td>
<td>243.86</td>
</tr>
<tr>
<td>Camburg (Saale)</td>
<td>0.79</td>
<td>9.66</td>
<td>0.64</td>
<td>17.05</td>
</tr>
<tr>
<td>Laucha (Unstrut)</td>
<td>0.65</td>
<td>5.26</td>
<td>0.68</td>
<td>10.34</td>
</tr>
<tr>
<td>Oberthau (Weisse Elster)</td>
<td>0.72</td>
<td>6.53</td>
<td>0.16</td>
<td>15.68</td>
</tr>
<tr>
<td>Piorau (Mulde)</td>
<td>0.67</td>
<td>24.34</td>
<td>0.09</td>
<td>63.32</td>
</tr>
<tr>
<td>Calbe (Saale)</td>
<td>0.77</td>
<td>25.21</td>
<td>0.62</td>
<td>44.05</td>
</tr>
<tr>
<td>Tangermünde (Elbe)</td>
<td>0.66</td>
<td>143.46</td>
<td>0.39</td>
<td>377.96</td>
</tr>
<tr>
<td>Wittenberge (Elbe)</td>
<td>0.57</td>
<td>167.82</td>
<td>0.38</td>
<td>400.95</td>
</tr>
<tr>
<td>Neu Darchau (Elbe)</td>
<td>0.59</td>
<td>171.56</td>
<td>0.32</td>
<td>421.99</td>
</tr>
</tbody>
</table>
partially because the authorities that are in charge of the operation of such structures do not always provide all necessary information on the regulation rules. Also, rules may be complex, they may change over time, or in particular cases they may not even be followed at all. Because of this, we have not taken this issue into consideration in the examples presented here. However, with more and better data, we expect that further improvement may well be possible. This issue should be addressed in further studies. In section 2 we already mentioned how the simulation of vertical flow processes in LISFLOOD is always in a downward direction, and that this may have implications for the use of the model under dry conditions, or in cases with a profound influence of deep groundwater systems. Current work with the model has been largely confined to humid, west- and central-European catchments. Since LISFLOOD was developed with the aim of being applicable throughout Europe, future studies should investigate the performance of the model under drier (e.g. Mediterranean) conditions. Also, most current applications have a strong focus on the simulation of floods; more work is needed to evaluate the performance of the model during low-flow periods. Both issues will be addressed in upcoming studies.

Acknowledgements
Over the years of LISFLOOD development, many people have contributed to the development and testing of the code. We would like to acknowledge in particular the contributions of Dr. Francesca Somma, Mr. Gian Franchello, Dr. Willem van Deursen and Mr. Cees Wesseling. Furthermore we would like to thank all our present and past colleagues at the JRC Floods group for their continuous feedback on the model.

References


