E.T.S.I.Agrónomos Universidad Politécnica de Madrid

Departamento de Ingeniería Rural

MODELLING OF TEMPISQUE RIVER WATERSHED IN NW COSTA RICA, USING TRANSFER FUNCTION MODELS

Carlos García de Vinuesa Llamas

Directores:
Dr. Luis Juanas.
Departamento de Ingeniería Rural
UPM

Dr. Marnik Vanclooster
Earth and Life Institute-Université catholique de Louvain

ABRIL 2014

ACKNOWLEDGMENTS

I would like to specially thank my advisor, Marnik Vanclooster for giving me the opportunity to participate in this project, for all of his work, and for being patient with me.

Thanks also go Université catholique de Louvain Earth and Life Institute for giving me generous office space and computer resources that were a great help in working on the project.

Warm thanks go to all researchers of Earth Life Institute for welcoming me like one of them and always being willing to lend a hand.

I would like to thank Alice Alonso from University of Florida for sharing her work with me and supporting me during the execution of the project. Her help and orientation through our Skype meeting gave me courage to follow through with the task.

Finally, thanks go to Luis Juanas of Universidad Politécnica de Madrid for supporting me from Madrid and supervising the work.

CAUTIONARY SECTION

MODELLING OF TEMPISQUE RIVER WATERSHED IN NW COSTA RICA, USING TRANSFER FUNCTION MODELS

1. Introduction (Overall background, Hypothesis and Objectives)

The Arenal Tempisque irrigation program in Costa Rica provides several water derived benefits for the economics of the country. But this programme also triggered an unexpected increase in water withdrawal from groundwater and rivers of the basin, producing changes in the hydro period and decrease of discharge intensity of the rivers.

For proposing appropriate sustainable water management in the basin, a thorough understanding of the fluxes (rainfall-runoff fluxes, surface water fluxes, surface-groundwater fluxes …) in the catchment is needed. Water fluxes between different compartments of the Arenal-Tempisque hydrosystems can be quantified through hydrological modelling.

Hydrological fluxes into a complex hydro-system like the Arenal-Tempisque catchment can be conceptualized by means of a hydrological network. A hydrological network consists of a set of nodes and links. Each node corresponds to a location for which a time dynamic hydrologic attribute (precipitation, discharge, groundwater depth) is modelled. The links allow modelling the fluxes between the nodes.

Based on the availability of flow observations in the catchment (rainfall stations, discharge stations, groundwater stations …), we aim implementing conceptual based hydrological transfer function models for modelling the links of a hydrological network. The transfer function fitting parameters will be interpreted as indicators of connectivity between the different nodes of the network, while the transfer function parameters themselves will be interpreted, for the connected nodes, as travel time probability distributions, measuring the velocity with which the hydrological signal propagates through the network.

The specific objective of this work is to identify the structure and type of the conceptual hydrological model that can be used in such a context, based on the observations of discharge and precipitation at some different locations in the Arenal-Tempisque catchment.

2. LITERATURE REVIEW:

2.1 Hydrological modelling

2.1.2 The rational for hydrological modelling

There are many different reasons why rainfall-runoff models are developed in hydrology, though the most important reason is the fact that hydrological measurement techniques are limited for assessing the rainfall-runoff relationships (Beven, 2001).

Indeed, despite all the experimental and technological advances made for exploring the subsurface like in the area of remote sensing, ground-probing radar, etc., our knowledge of the functioning of the hydrological system, in particular in relation to underground processes is still very limited.

The patterns of water movement in hydrological systems are very complex. This complexity at the scale of practical interest does not allow reproduce the flow processes with accuracy. If we rely on the current measurement techniques, a large part of the hydrosystem is still inaccessible. This is particularly the case for the flow in soil and groundwater systems. Since soil and groundwater determines the rainfall-runoff relationships, we should be able to integrate these subsystems in rainfall-runoff assessments, implying the development of alternative assessment techniques that are based on hydrological modelling.

Hence rainfall-runoff modelling allows the quantitative assessment of the rainfall-runoff relationships and plays an important role in water resources assessment, water resources management and water resources decision-making. With hydrological models, synthetic sequences of hydrologic data can be generated supporting the facility design or forecasting of the behavior of water systems. Some of the practical applications are the design of engineered channels, flood forecasting, the assessment of the impact of effluents on water quality, the prediction of pollution incidents, the evaluation of the potential impacts of climate and land use change on hydrology, etc.

Sometimes we do not need develop the complex predictive model with full details, in fact many rainfall-runoff models are very simple and the results are successful.

2.1.3 Approaches developed in hydrological modelling

An enormous range of hydrological models have been produced. These developments were boosted by the variety of uses and applications of hydrological models, the rapid increase in scientific understanding of the functioning of the hydrological cycle, the technological developments for data collection system and computing technology. Wheater et al., (1993) and Wagener et al., (2004) have categorized the models into the following three categories.

Empirical (also called metric, data-based or black-box) models.

These models are based primarily on observational time series and the main aim is to characterize the flow response largely on the watershed of these data, generally through some form of statistical estimation or optimization.

These models do not include any prior knowledge about catchment behavior and flow processes (Wagener et al., 2004). They are usually spatially lumped and purely based on the relationship of a series of input to a series of output data.

The rational method model, regressive time series models, Artificial Neural Networks (ANN) and empirical unit hydrograph methods are examples of such models (Beven, 2001).

Conceptual (also called parametric or gray-box) model.

These models are the most common class of hydrological models (Wheater, 2008) and vary considerably in complexity.

Conceptual models are normally based on the representation of internal storages. These storages are filled through fluxes such as rainfall and emptied through evapotranspiration, runoff, drainage, etc (Wagener et al., 2004).

The essential feature of these models type, is that the model structure is specified a priori, based on the modeller’s perception of the relative importance of the component processes at work in the catchment.

These models have three distinguishing characteristics (Wheater et al., 2008): (a) based on the modeller’s understanding of the hydrological system, their structure is specified by the user prior to their use; (b) hydrological properties of the catchment such as size of the storages elements or the distribution of flow between them are described by parameters; and (c) as some processes are usually aggregated into a single parameter, these parameters have no direct, physical meaning and cannot be derived from field measurement.

Since some model parameters have no direct physical meaning, model parameters have to be assessed through the ‘calibration’, using available rainfall and flow data.

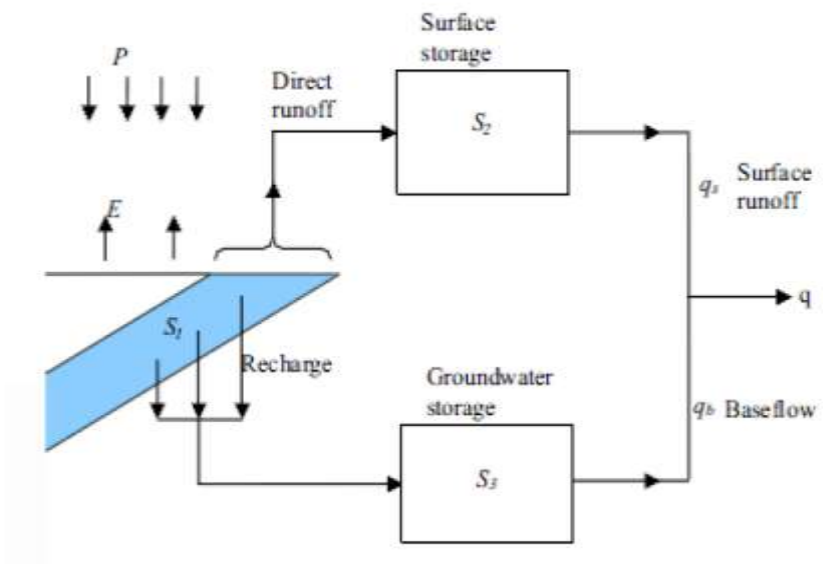


Figure 1: Representation of a conceptual model.

Physical-Based Models (also called mechanistic or white-box).

These models are based on physical principles as conservation of mass, momentum and energy. They are explicitly founded on the best available understanding of the physics of the hydrological processes (Wheater, 2008).

These models represent the functioning of components of the hydrosystems using classical, mathematical-physics form, such as differential equations for describing continuum mechanics. These equations are typically solved in an approximate manner via finite difference or finite element spatio-temporal discretization methods.

Physical based models often suffer from extreme data demand, scale-related problems (the measurement scales differ from process scales) and over-parameterization (Wagener et al., 2004).

The high-dimensional parameterization makes objective optimization and calibration virtually impossible, since the model is normally so over-parameterized that the parameters value cannot be uniquely identified and estimated against the available data (Young, 2001).

They are particularly appropriate when a high level of spatial discretization is important, e.g. soil erosion or pollution studies (Wagener et al., 2004). A clear example of this category of models is the Système Hydrologique Européen (SHE) model (Abott et al., 1986).

2.1.4 Some conceptual hydrological models

In this section we will review some conceptual hydrological models, as these model types will be used for modelling rainfall-runoff in the Tempisque case study.

The Linear Store Transfer Function model (LSTF)

The LSTF model uses a general linear model to model output (e.g. run-off in a given catchment) in terms of the input (e.g. storage in a catchment).

A linear store is a model element for which the predicted output, Q [L^3 T^{-1}], is directly proportional to the storage, S [L^3] (Beven, 2001).

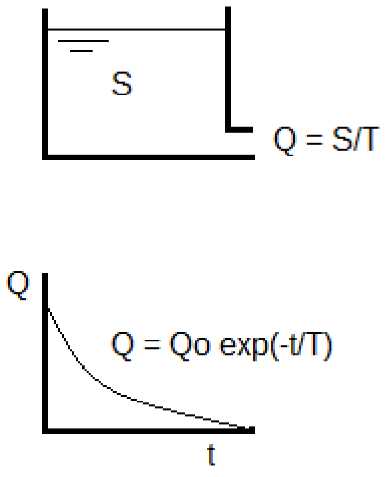


Figure 2: The linear store.

We assume:

$$Q = S/T \qquad (1)$$

where T [T] is a parameter equivalent to the mean residence time of the store. The linear store is physically equivalent to a straight-sided bucket with a hole in the bottom, allowing the storage in the bucket to escape as an output Q.

In continuous time, the impulse response or transfer function of the linear store is expressed as

$$Q_t = u/T \exp\{-(t - t_0\}/T\} \qquad (2)$$

where u [L^3] is storage input, i.e. the effective rainfall volume. It has the form of an initial step rise followed by an exponential decline in the outflow as is shown in the figure above.

The resulting models may, however, have useful mechanistic interpretations. For example, evaluations of some catchment transfer functions have frequently suggested that a parallel model structure is appropriate, with a proportion of the runoff being routed through a fast pathway and the remainder through a slow pathway (Beven, 2001).

The IHACRES Model

The IHACRES model (Identification of unit Hydrographs and Component flows from Rainfall, Evaporation and Streamflow data) of Jakeman et al. (1990) derives from the work of Young (1975) and Whitehead et al. (1979). The model avoids the problem of hydrograph separation in classical unit hydrograph models.

The IHACRES is a transfer function model which relates input (rainfall) to an output (discharge).

Beven (2001) suggested that linear transfer functions may be appropriate to model rainfall discharge relationships in catchment if an appropriate nonlinear filter allows the calculation of the effective rainfall in terms of total rainfall.

The IHACRES model uses a particular set of nonlinear functions to filter the rainfall, it introduces a soil storage variable and also, for a longer period simulation, uses temperature as an index evapotranspiration, to produce an effective rainfall. The effective rainfall is then related to total discharge using a generalized linear transfer function.

This model has been applied in different applications using several forms of the rainfall filter, particularly in a wide variety of catchments including some catchments subjected to significant snowmelt inputs. The model has also been used in the prediction studies of the impacts of climate change on catchment hydrology.

The model further provides two parallel coupled linear storage functions, one for fast flow pathway and one for slower flow pathway. The fast flow will provide the major part of the predicted storm hydrograph, the slower pathway the major part of the recession discharge between storm periods.

The main advantage of the IHACRES approach is that the data allows to suggest a part of the form of the transfer function (e.g. the distribution between the slow and fast store is parameterized). Hence the model structure is not fully fixed beforehand.

The Sacramento Model

The Sacramento Soil Moisture Accounting (SAC-SMA) model represents the moisture distribution in a physically realistic manner within hypothetical zones of a soil column, which can be split in two conceptual layers.

The components of the SAC-SMA are tension water and free water; the fluxes for each soil column are surface flow, lateral drainage, evapotranspiration (ET), and vertical drainage (percolation).

Rain falling on the soil column first encounters the upper zone. Here, rain falling on any impervious areas generates impervious area runoff, while rain falling on the non-impervious areas of the basin first encounters the upper tension water storage. After filling this reservoir, excess soil water enters the upper zone free water. Water in this free water storage can percolate into the lower zone storages or flow out as interflow. If the upper zone free water fills completely, then excess soil water flows out as surface runoff. Most percolated water flows into the lower zone tension water storage, although some can go directly to free water storages in the lower zone. Upon filling the lower zone tension water storage, all soil water moves into the two lower zone free water storages. These two free water storages generate fast and slow responding base flow. The combination of these two base flows is designed to model a variety of hydrograph recessions.

The SAC-SMA model is very often represented graphically as illustrated in the Figure 3.

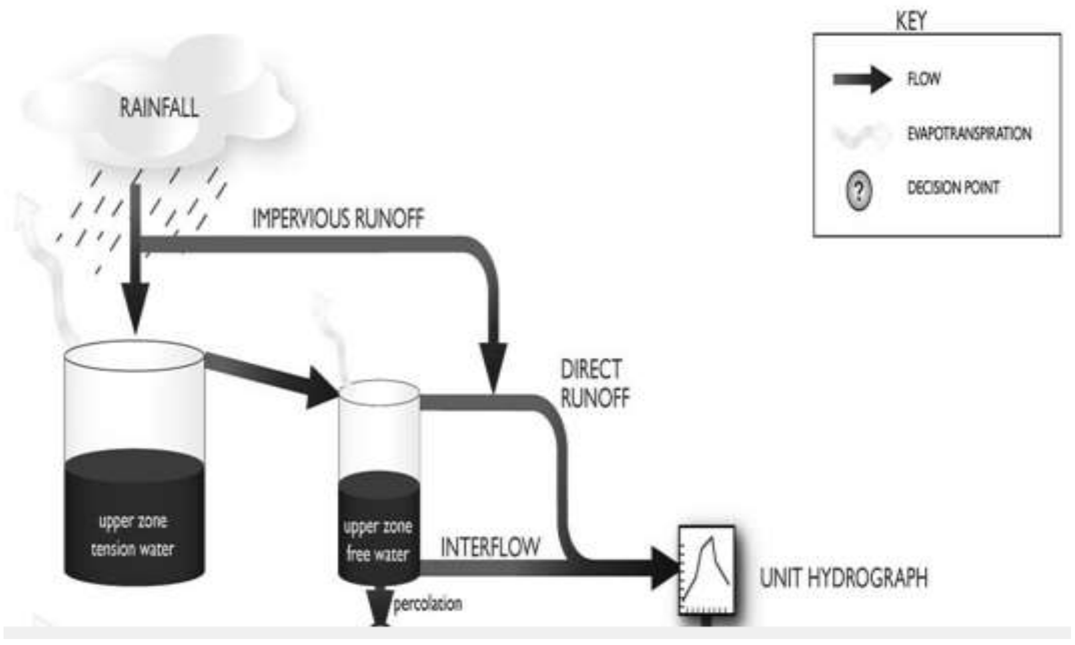


Figure 3: Representation of SAC-SMA model.

The GR4J Model

The GR4J model developed in France is a four parameters lumped rainfall-runoff model. Its structure is similar to that of many conceptual type models. This model was developed following an empirical approach, i.e. without a priori ideas on the rainfall-runoff transformation, but trying to find the model structure that performs best on a large set of hydro-climatic conditions.

The GR4J model takes the rainfall and potential evapotranspiration over each sub-basin as inputs and computes the discharge at their downstream end.

The discharge production module is based on (Andréssian et al., 2006):

- Net rainfall and potential evapotranspiration which are determined with a zero capacity interception store.
- Soil moisture accounting (SMA) store to determine: (i) the part of raw rainfall that will become effective rainfall; and (ii) the actual evapotranspiration.
- A water-exchange function that can simulate import or export of water from/to the subsurface outside of the catchment. It acts on the two components simulated by the transfer module.

In the transfer function, the transfer production module is based on (Andréssian et al., 2006):

- A percolation from the SMA store.
- A constant volumetric split of effective rainfall into a direct flow component (10%) and an indirect flow component (90%).
- Two unit hydrographs (UH), each one acting on one flow component.
- A nonlinear routing store that routes the indirect flow component.

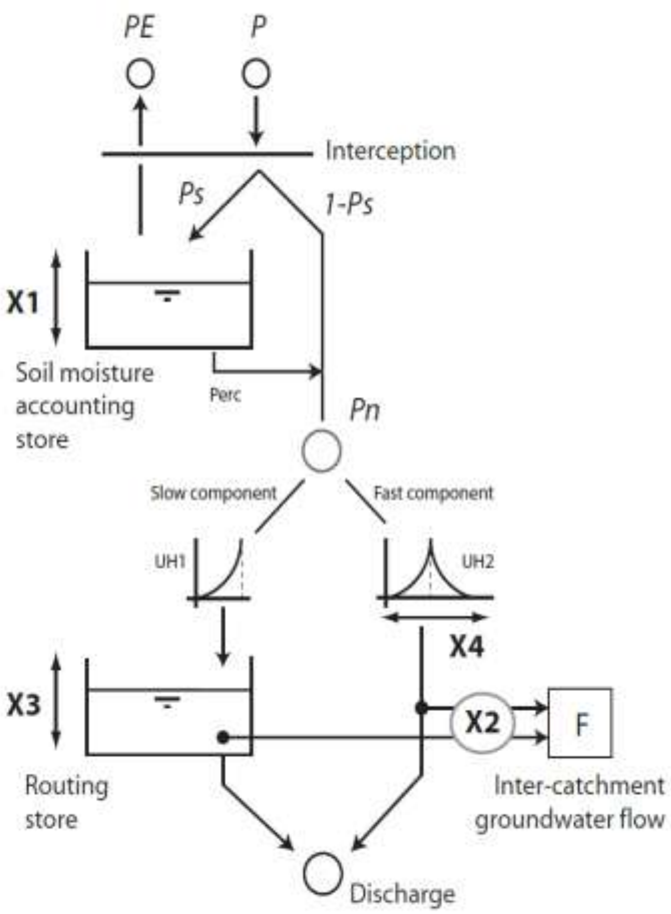


Figure 4: GR4J rainfall-runoff model scheme.

The PDM Model

The Probability Distributed Moisture (PDM) model is a conceptual model, which uses a distribution of soil moisture storage capacities for soil moisture accounting and a number of linear or non-linear reservoirs for the routing component (Moore, 2007).

One of the main reasons for introducing a distribution of storages was to make the calibration problem easier, because it is possible obtain smoother response surface in comparison with other models.

The PDM split rainfall into direct runoff, groundwater recharge and soil moisture storage. A probability-distributed soil moisture storage component is used to separate between direct runoff and subsurface runoff. Direct runoff is routed through the 'fast-response' surface storage representing channel and other surface flow mechanisms to provide surface runoff. This employs a two linear reservoir cascade. Groundwater recharge from soil water drainage is routed through subsurface groundwater storage which entails a 'slow response' system representing the groundwater and other base flow components of total runoff.

The main advantages of the PDM model are it´s analytical and computational simplicity. This model is capable to provide good simulations of observed discharges in many applications. The distribution of conceptual storages can be interpreted as an appropriate realistic representation of the functioning of the catchment.

The basic idea of the PDM model is illustrated in the next figure.

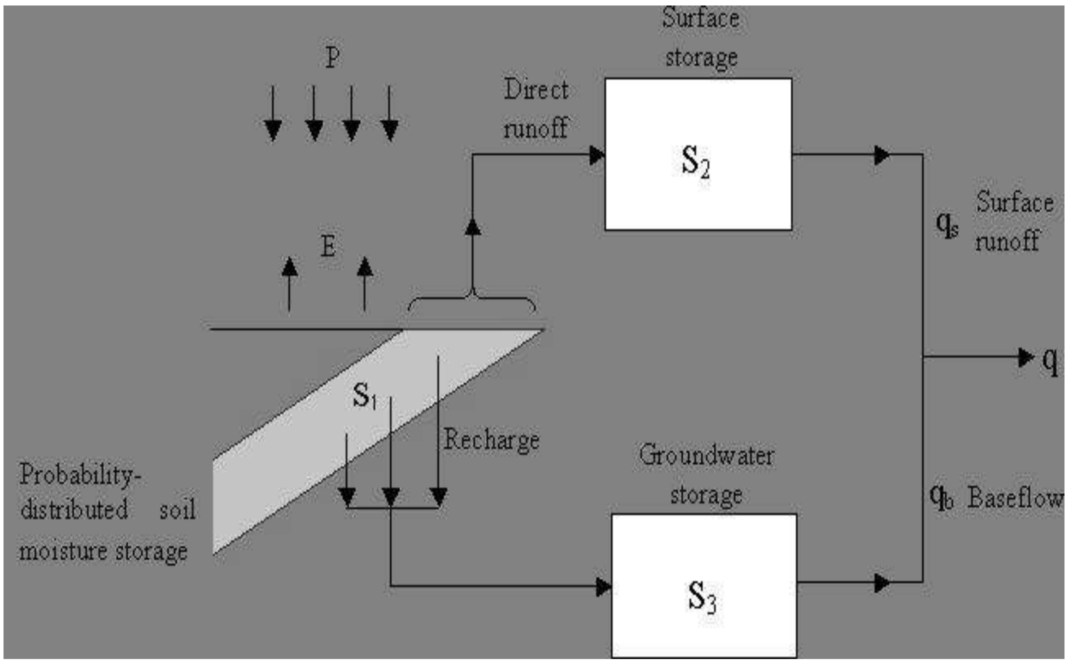


Figure 5: Structure of the probability distributed model (PDM).

The TOPMODEL

TOPMODEL is a hydrologic model that bases its distributed predictions on an analysis of basin topography.

The model may be seen as a product of two objectives. One is the development of a pragmatic and practical forecasting and continuous simulation model. The other is the development of a theoretical framework within which perceived hydrological processes, issues of scale and realism and model procedures may be researched (Beven, 2001). This model allows identifying the patterns of response in a catchment through a simple approach.

TOPMODEL split the basins into a set of subbasins and use the practice of calculating actual evapotranspiration, as a function of potencial evapotranspiration and root zone moisture storage.

Surface runoff is computed based on variable saturated areas, and the subsurface flow is calculated using an exponential function of water content in the saturated zone.

Beven and Kirkby (1979) applied the TOPMODEL to simulate small catchments in the UK, and they demonstrated that it is possible to get reasonable results with a minimum of calibration of parameter values.

2.2 The Unit Hydrograph model

2.2.1 Definition

The unit hydrograph can be defined as the hydrograph of unit volume of storm runoff produced by a unit volume of uniform intensity excess rainfall over a unit period, distributed uniformly in a catchment. In the unit hydrograph concept, the catchment behaves like a linear time-invariant system.

The concept of the unit hydrograph was introduced by Sherman in 1932 as a method for predicting floods from available rainfall-discharge data.

The traditional unit hydrograph technique proposes specific forms on the linear transfer function, e.g. the triangular transfer function or the transfer function that corresponds to the Nash cascade (figure 6). Some specific implementations are reviewed below.

Nash Method

In 1958, Nash presented a conceptual model in which he described a mathematical model of a river basin, that simulated the response of the basin to a unit impulse of rainfall excess. To explain the theory, Nash utilized the effect of "lamination" of an effective rainfall having the depth of 1 mm and duration that tends towards zero. This lamination is performed by a cascade of *n* linear reservoirs having identical storage coefficients *K.* Nash posed his model as an instantaneous unit hydrograph (IUH).

The resulting mathematical form for the unit hydrograph is equivalent to the gamma distribution that he expressed as:

$$u(t) = \frac{1}{k\Gamma(n)}(\frac{t}{k})^{n-1}\exp{(-\frac{t}{k})} \qquad (3)$$

where u(t) is the ordinate of IUH (hour^{-1}), t is the sampling time interval (hour), n and k are the parameters of the Nash model, in which n is the number of linear reservoirs, and k is the storage coefficient (hour), Γ (n) is the gamma function.

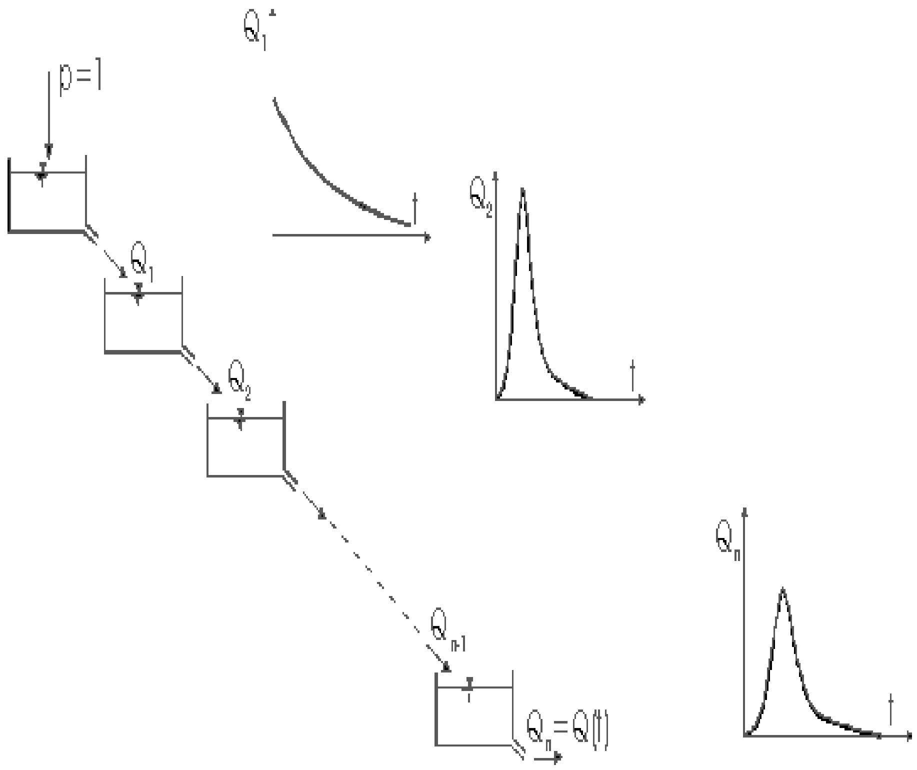


Figure 6: Nash cascade of n linear stores in series.

Snyder Method

This method developed in 1938 is called the synthetic unit hydrograph of Snyder. It is based on relationships found between three characteristics of a standard unit hydrograph and descriptors of basin morphology. Snyder standardized the unit hydrograph defining the characteristics of effective rainfall duration,(t_r(hours)); the peak direct runoff rate (q_p(m^3/s)); and the basin lag time (t_l(hours)) (Ramirez, 2000).

The specific effective rainfall duration of a standard unit hydrograph can be defined by the following relationship with basin lag time:

$$t_l=5.5t_r \quad (h) \qquad (4)$$

Considering a standard unit hydrograph, the basin lag time, and the peak discharge, can be expressed as follows:

$$t_l=C_1C_t(LL_c)^{0.3} \quad (h) \qquad (5)$$

$$q_p=(C_2C_PA)/t_l \quad (m^3/s) \qquad (6)$$

where C_t (usually ranging from 1.8 – 2.2) is a non-dimensional coefficient derived from gauged watersheds in the same region, and represents variations in watershed slopes and storage characteristics and C_1 = 0.75 (1.0 for English units). L is the length of the main stream from the outlet to the catchment boundary in miles, L_c is the distance from the outlet to a point on the stream nearest to the centroid of the watershed in miles. C_p is another non-dimensional coefficient derived from gauged watersheds in the area, and represents the effects of retention and storage, C_2 = 2.75 (640 for English units), and A is the basin area in km^2.

From the derived unit hydrograph of the watershed, values of its associated effective duration t_r in hours, its basin lag t_{lr} in hours, and its peak discharge q_{pr} in m^3/s are obtained. If $t_{lr} = 5.5t_r$, then the derived unit hydrograph is a standard unit hydrograph and is possible get the values using the equations shown above. If t_{lr} is quite different from $5.5t_r$, the basin lag of the standard unit hydrograph for the basin is adjusted as follows:

$$t_l=t_{lr} + (t_r-t_R)/4 \qquad (7)$$

Snyder's method is applicable to fairly large catchments only, e.g., 100-500 km^2 (Taylor and Schwarz, 1952; Gray, 1961).

Unit Hydrograph of Clark

This method was proposed by Clark in 1945 and has been applied in a large range of hydrological studies in the 50´s and 60´s. The method is based on the surface distribution of watershed between isochrones lines. The method calculates the production of effective runoff in every surface area belonging to an isochrone class, and the transfer towards the outlet using a linear transfer function model.

This method assumes that the considered basin operate as a reservoir. An increase in the inlet flow of a reservoir produces effect in the outlet flow, but the outlet flow will be damped and delayed as compared to the inlet flows.

The simple version is to consider a linear reservoir, that is, as we have explained before, a linear relationship between the storage volume in the reservoir and the outlet flow.

$$S = Q\,R \qquad (8)$$

Where

S, is the storage volume (m^3);

Q, the outlet flow in reservoir (m^3/s);

R (s), the proportionality constant.

For a given time increment (Δ_t):

$$V_{in} - V_{out} = \Delta S \qquad (9)$$

where, V_{in}, is the inlet flow volume (m^3) in a $\Delta_{t.}$;

V_{out} (m^3), is the outlet flow volume in the same Δ_t;

ΔS (m^3) is the storage volume variation in the Δ_t.

If we divide the last equation by Δ_t, we obtain:

$$I - Q = \Delta S / \Delta_t \qquad (10)$$

Where

I (m^3/s) is the mean inlet flow rate in the Δ_t;

Q (m^3/s) is the mean outlet flow rate in the $\Delta_{t.}$

Soil Conservation Service (SCS) Method

The Soil Conservation Service (SCS) of US, now Natural Resources Conservation Service (NRCS) developed in 1957 a Dimensionless Unit Hydrograph (DUH) based on the analysis of large number of watersheds.

In this method all the hydrograph ordinates are given by ratios between instantaneous discharge and peak discharge and between time and time-to-peak, as illustrated in the next figure. The unit hydrograph peak discharge also is directly related to the time-to-peak from consideration of the volume of the unit hydrograph. This is best illustrated for the SCS dimensionless, triangular unit hydrograph shown in the figure.

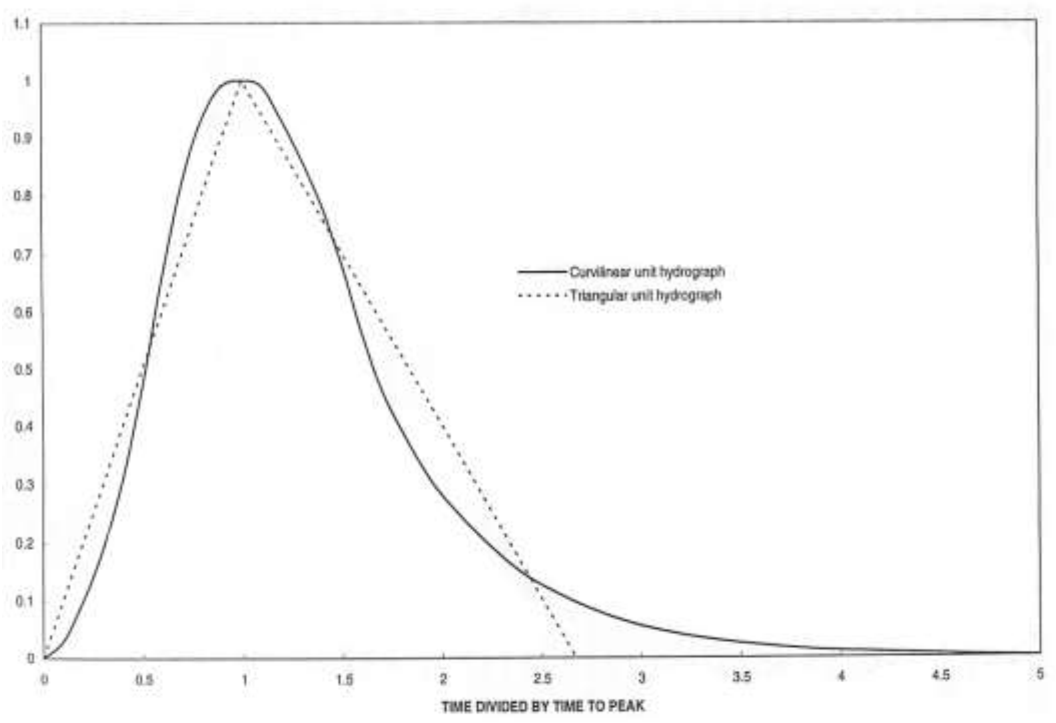


Figure 7: Soil Conservation Service dimensionless curvilinear unit hydrograph and equivalent triangular unit hydrograph (Melching et al., 1997).

The volume of runoff under the SCS dimensionless, triangular unit hydrograph is given by

$$V = 1.800\ q_p(T_p + T_r) \qquad (11)$$

where

q_p (ft^3/s) is unit hydrograph peak discharge;

T_p (hours) is the time-to-peak discharge in hours; and

T_r (hours) is the time of recession, which is equal to $1.67T_p$ for the SCS dimensionless, triangular unit hydrograph.

Therefore, the volume is

$$V = 1.800\ q_p(T_p + 1.67T_p) = 4.800\ q_pT_p \qquad (12)$$

and the triangular unit hydrograph peak discharge is

$$q_p = 484\ A/T_p \qquad (13)$$

2.2.2 Probability Density Function approach

The unit hydrograph of a watershed under the assumption of linear system may be defined as the unit impulse response function (or Dirac delta) of a linear system.

Physically, the unit impulse can be described as an input or excitation of unit magnitude, imposed either suddenly and lasting a very short time over a wide area, or imposed locally and acting over a very small distance or area but in a steady fashion (Brutsaert, 2005).

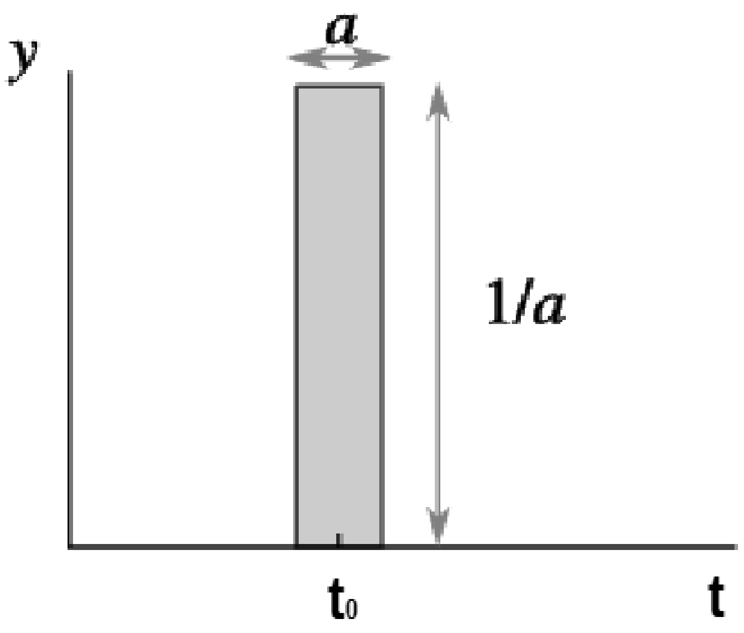


Figure 8: Example of an impulse function of uniform intensity 1/a and duration a.

In general, the delta function is often expressed as

$$\delta(t - \mathrm{to}) = \begin{cases} 0 & for\ t \neq to \\ \infty & for\ t = to \end{cases} \qquad (14)$$

$$\int_{-\infty}^{+\infty} \delta(t - \mathrm{to})dt = 1 \qquad (15)$$

Although this definition cannot be taken literally. It must be interpreted as suggestive of the limiting process involved, because $\delta(t - \mathrm{to})$ is not continuous and not differentiable at t=t_0. A better way to define the delta function is in the following integral form

$$\int_{-\infty}^{+\infty} \delta(t - \mathrm{to})f(t)dt = f(to) \qquad (16)$$

in which f(t) is a continuous and smooth function.

The transformation of input into output is called the response of a system. The unit response of a linear system u=u(t) is its response to the unit impulse function δ(t). If these characteristics are invariant in time or space, the response is u(t_0 - t), when the input is δ(t_0 - t). Due to that we have a linear system, the response will be x(t)u(t_0 - t) when the input is x(t)δ(t_0 - t). Multiplying the response and the input by dt and doing the integration of both, leads to the conclusion that when the input is x(t), the response or output of the system is given by

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)u(t-\tau)d\tau \qquad (17)$$

where τ is a dummy variable of integration and t is the time. This operation is called the convolution integral.

The limits of this equation mean that the output from the system is affected by input values of x(t). In hydrologic applications, the upper limit of the before integral should be the time (t), therefore one can write

$$y(t) = \int_{-\infty}^{t} x(\tau)u(t-\tau)d\tau \qquad (18)$$

which represent the output from a system with a memory going back to -∞.

A big numbers of studies have been conducted to develop unit hydrographs using probability density functions (PDFs), because the unit hydrographs satisfies all the properties of a probability distribution function. Changing the parameter values of the PDFs, is possible to obtain different shapes of UHs and the shape characterizing the PDFs contains many information about the processes that drive a water particle thorough the catchment (Rinaldo et al., 1991), (D'Odorico & Rigon 2003).

The travel time of water can be defined as the time spent by a water particle in a catchment from its entrance as rainfall until its passage on a point where streamflow is measured (Botter et al., 2011).

A probability density function can represent a travel time which consider the hydrologic process as stochastic. This PDF provide information about the hydrological response of the catchment against rainfall.

2.3 Corrections in the hydrologic attributes for identifying rainfall-runoff transfer functions

The quality of input data largely determines the success of a hydrological model. Recently several techniques have been developed to improve the estimation of input data of hydrological models. We will focus here on the rainfall-discharge and discharge data.

Precipitation data

Precipitation is the main input in a hydrological modeling. Rainfall can be measured by means of raingauges.

However precipitation measurements are affected by systematic errors, which lead to an underestimation of actual precipitation. In particular, this error depends on the design of the raingauge in relation to wind conditions at the site and rainfall intensities. The best design is thought to be a raingauge with the orifice set at ground level and surrounded by an anti-splash grid, but this is not always practical, particularly in environments with frequent snow.

The wind speed is undoubtedly the most important factor determining the systematics errors. However there are others factors involved, as the error due to wetting loss on the internal walls of the collector, and a wetting loss when it is emptied (according to the WMO-8, 2008 about 2-15% in summer and 1-8% in winter). Also there are errors due to evaporation from the container. This is especially important in hot climates (according to the WMO-8, 2008 up to 4%). In addition, there are the trace precipitation error (according to Sugiura et al. 2003 about 6-130%), the systematic mechanical errors, the errors due to in- and out-splashing of water (according to the WMO-8, 2008 about 1-2 %) and errors due to blowing and drifting snow (Wagner, 2009).

The systematic mechanical error mentioned is directly relational with the high rainfall intensities, especially in some types of recording raingauge, such as the tipping bucket. In this case the bucket requires a specific calibration.

For practical application, the rainfall measurement errors should be minimized. Several techniques can be used out to correct the different types of error. We will expose some of them.

The trace precipitation error can be corrected according to Yang et al. (2001) by adding 0.1 mm in the daily rain measure. However Woo and Steer (1979) designed a method of measuring trace rainfall in the Canadian high Arctic and determined a mean rate of 0.01 mm per day.

The error due to evaporation can be estimated as follows according to the WMO Guide to meteorological practices (No-168, 1994):

$$\Delta Pe = i_e \tau_e \qquad (19)$$

where

i_e= evaporation intensity [mm/day]

τ_e = duration of evaporation (fractions of a day)

The value of i_e depends on the construction, material and color of the gauge, on the amount and form of precipitation, on the saturation deficit of the air and on wind speed at the level of the gauge rim during evaporation. The theoretical estimation of the evaporation intensity (i_e) is difficult because of the complex configuration of a precipitation gauge (Wagner, 2009).

Several methods have been developed to avoid the wind effects in the rain measure. Some of them are the method developed by The World Meteorological Organization (WMO), the correction method according to Sevruk (2004), the method of Richter (1995) or the method according to Chang and Flannery (1998).

Raingauges are also suffering from a small spatial support. To improve the spatial support of the measurement rainfall radar measurement are nowadays proposed. This device has led to a much greater appreciation of the temporal and spatial variation of rainfall intensities than was previously available from raingauge measurements alone (Beven, 2001).

Discharge data

The availability of discharge data is essential for the model calibration process. Normally the discharge data are available only in few locations in a catchment.

There are many different ways of measuring discharges. Usually it is difficult to make a direct flow measurement, except for very small flows. The level of water in a channel is directly proportional to discharge, and this water level is relatively easy to measure. Yet the stage-discharge relationship depends on the governing flow regime and is not unique for any cross sections.

In a well-maintained weir or flume structure, the relationship between stage and discharge is determined by well-established hydraulic laws. In these conditions, the stage discharge conversion can be accurate (better than 5 percent, Beven, 2001). But sometimes accuracy can be worse than this, for example when if the structure is overtopped in a high flow.

When there are extreme floods, the water-level measuring device may itself be washed away. In this case the flow can be estimated by means of the slope-area method. With this method, the cross-sectional area of the flow and the slope of the water surface are estimated from the trash lines indicating the maximum extent of the flow, and a uniform flow roughness equation used to determine an average velocity.

In this case, however, the flow may be non-uniform, highly turbulent, and with high sediment load. Hence the effective roughness coefficient and cross-sectional area, and hence he average velocity and discharge will be very uncertain (Beven, 2001).

The modeller often forgets these potential errors, normally they use the raw discharge data. However, if a model is calibrated using data that are in error, then the effective parameter value will be affected. As a consequence these errors in the parameters will affect the predictions for other periods depending on these parameters.

Some errors may be avoided doing simple checks as:

- Calculate the total volumes of rainfall and runoff for different periods in the record, choosing periods separated by similar low flows where possible so that the calculated volumes are not greatly affected by recession discharge.
- If more than one discharge gauge or raingauge are available, check for consistency between the gauges.
- Check for any obvious signs that infilling of missing data has taken place.

2.4 Model Calibration

Hydrological models allow to predict run-off in terms of rainfall, and hence to support water resources management and engineering, if, and on only if, the model parameters of the hydrological model are known. In many cases, hydrological models are not known a priori and needs to be estimated based on limited observations of precipitation and discharge data. This process of parameter estimation is often done during the model calibration phase or the model inversion. Inversion refers to the process where model parameters are estimated from discharge and rainfall data. The model inversion process is in contrast to the direct modelling process, where run-off data are estimated from rainfall data and model parameters.

The "inverse problem" consists in using the results of actual observations to infer the values of the parameters characterizing the system under investigation. The technique is based on the calibrating selected parameters using an iterative process of three basic steps: (i) parameter perturbation; (ii) forward modelling; and (iii) objective functions evaluations (Ritter et al., 2003).

Inverse problems may be difficult to solve for at least two different reasons: (1) different values of the model parameters may be consistent with the data, and (2) discovering the values of the model parameters may require the exploration of a huge parameter space.

Furthermore inverse problems are most difficult to solve than direct problems because the problem is the minimization of a non-linear object function which implies the solution of the direct problem for many possible parameter realizations. This can be done using iterative methods such as Newton's Method or The Gauss-Newton Iteration. The optimal parameter set is the one, which produces the optimal objective function.

We explain below two methods, the simple "trial and error" method, and the method that we will use in this project, Global Multilevel Coordinate Search method.

2.4.1 The "trial and error" procedure

The trial and error is a subjective but simple method of solving inverse problems. With this method, parameters are tuned progressively from an initial set towards the optimal set in terms of some loosely defined object function. The tuning, however, is unsystematic and often subjective. Though very simple, the method should be avoided since the parameters may be situated within local optima and results may be largely influenced by the subjective initial choices of the model parameters. Also, the model does not allow assessing the quality of the obtained parameters.

2.4.2 Global Multilevel Coordinate Search

The GMCS method belongs to a category of increasingly popular global optimization algorithms that are designed to get over the complex topography and multimodality of the multidimensional nonlinear objective functions without requiring excessive computing resources (Lambot et al. 2002).

Algorithms for solving global minimization problems can be classified into heuristic methods that find the global minimum only with high probability like, e.g., Simulated Annealing or Genetic algorithms, and stochastic methods that guarantee to find a global optimum with a required accuracy. The GMCS algorithm is an intermediate between both methods.

GMCS does not need to calculate derivatives of the objective function, causing it to be very insensitive to possible discontinuity of the objective function. The task of global optimization is to find a solution for with the objective function obtains its smallest value, the global minimum. When the objective function has a huge number of local minima, local optimization techniques are likely to get stuck before the global minimum is reached, and some kind of global search is needed to find the global minimum with some reliability (Huyer, W. et Neumaier, A. 1999).

Basically, using the GMCS, the parameter search space is split into smaller 'boxes'. Each box is characterized by its midpoint, whose function value is known. A box can be split into smaller ones. As a rough measure of the numbers of times a box has been split, and a determined level is assigned to each box (Ritter et al. 2003 and Ritter et al. 2004). The algorithm starts with the boxes at the lowest level, and it is the global part of the algorithm. When the level of a box reaches a specified maximum value, the box is considered too small for follow the split and thus, the local search changes.

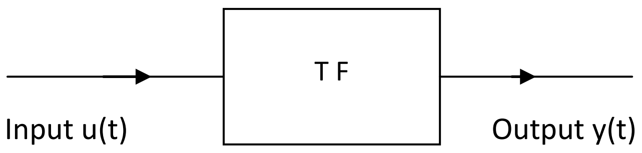
The GMCS is a good alternative to other optimization algorithms: initial values of the parameters to be optimized are not needed and it is very robust, because it can deal with discontinuous nonlinear multimodal objective functions (Ritter et al. 2003).

3. Modelling theory

3.1 The transfer function models

We use a transfer function model to describe the water transport through the hydrological network. A transfer function is a mathematical representation, in terms of spatial or temporal frequency, of the relation between the input and output of a linear time-invariant system (LTI).

The transfer function can be represented by the follow scheme:



where u(t) would be in this study the rainfall and y(t) the flow.

Mathematically, a transfer function in the Laplace domain can be defined as the quotient between the Laplace transform of the output (response function) and the Laplace transform of the input (excitation function, with the initial conditions of zero).

$$G(s) = \frac{Y(s)}{U(s)} + E(s) \qquad (20)$$

Where:

Y(s) is the Laplace transform of the output;

U(s) is the Laplace transform of the input;

G(s) is the Laplace domain Transfer function; and

E(s) is the Noise (can be considered zero).

In the discrete-time domain, the model equation is:

$$Y(z^{-1}) = G(z^{-1})U(z^{-1}) + E(z^{-1}) \qquad (21)$$

The transfer function in rainfall-runoff modelling allows the transformation of a time series of effective rainfalls into a series of predicted discharges.

3.1.1 Advantages

The main advantages of the transfer function models are the follow (Rabenstein, 1998):

- Is possible transpose the same procedure for setting up transfer function to a large number of technically important physical phenomena. There is no limitation in the number of spatial dimensions. No special system of coordinates is required and no special shape of the spatial domain is necessary.
- The inherent stability of a physical process is reflected by the stability of the corresponding transfer function, which can be checked by inspection of the denominator polynomial.
- The resulting discrete models are well suited for computer implementation since they require only addition, multiplication and delay elements and are free of implicit loops.
- As compared to numerical models, transfer functions do not employ a large number of calculations and can be easily implemented.

3.1.2 Disadvantages

The main limitation of transfer functions is that they can only be used for linear systems. While many of the concepts for state space modelling and analysis extend to nonlinear systems, there is no such analog for transfer functions and there are only limited extensions of many of the ideas to nonlinear systems.

3.2 Unit hydrographs

The theory of unit hydrograph was explained in the literature study. In this study we will use approaches linked to the unit hydrograph for modelling the hydrologic behaviour of the catchment. Since the unit hydrograph represents the probability that raindrops will reach the outlet, it reflects a probability density function. We can therefore use the unit hydrograph as a probability density based transfer function to model the rainfall-runoff process.

The unit hydrographs will therefore be represented using parametric Probability Density Functions (PDFs).

3.3 Probability Density Functions

The characterization of the time travel of water particles in the catchment will be achieved following the transfer function approach of Jury (Jury and Fluhler, 1992). This method uses a simple travel time probability density function (PDF) to relate an output time series to an input time series.

The simulated discharge is determined through the convolution between the rain and the probability density function as followed:

$$Q_{sim}(t) = \int_{-\infty}^{+\infty} f(t)P(t-\tau)dt \qquad (22)$$

where $Q_{sim}(t)$ is the simulated specific discharge (m^3/s); f(t) is the transfer function equal to the travel time probability density function (PDF) (1/s); $P(t-\tau)$ is the input time series, in our case the precipitation (m/s); and τ is the time lag between input water and water when it eventually enters the stream flow gauge. The transfer function f(t) measures the probability that a unit quantity of rain reaches the outlet in a time t + Δt.

Several probability density functions will be used in this study, concretely the Simple Transfer Function, Normal, Lognormal, Gamma, Beta and Weibull distribution. Brief descriptions about these PDFs are presented in this section.

3.3.1 Simple Transfer Function:

The Simple Transfer Function is a simple model for the probability density function curve.

This simple curve can be represented as follows:

$$H(t) = A - Bexp^{-\left(\frac{t}{T}\right)} \qquad (23)$$

where H(t) is the transfer function; t is the time; T the characteristic time; and A and B are the parameters.

3.3.2 Normal Probability Density Function

The normal or Gaussian density function has two parameters, μ and σ, which are the mean and standard deviation respectively. This distribution can be represented as follows:

$$f(t,\mu,\sigma) = \frac{1}{\sigma\sqrt{(2\pi)}} exp^{-\frac{(t-\mu)^2}{2\sigma^2}} \qquad (24)$$

The value of t is the time, must be always greater than 0; the value of μ can range between -∞ and +∞; and σ is always greater than 0.

The use of this distribution can be justified by means of the Central Limit Theorem; this states that, if a random variable is the sum of n random, not necessarily independent, variables, each with its own, not necessarily normal, density function with a finite mean and variance, then the density of this random variable tends to the normal function as n increases (Brutsaert, 2005).

3.3.3 Log-normal Probability Density Function

Many natural phenomena, which have a lower bound and exhibit positive skew, cannot be described well by the normal distribution, but in some cases their logarithms can. According to the Central Limit Theorem, this would be the case when the random variable is the product of n variables, each with its own arbitrary density function with a finite mean and variance (Brutsaert, 2005). The mathematical expression for the hydrograph using Log-normal PDF is presented following:

$$f(t,\mu,\sigma) = \frac{1}{t\sigma\sqrt(2\pi)} exp^{-\frac{(ln(t)-\mu)^2}{2\sigma^2}} \qquad (25)$$

The notations μ and σ are the mean and standard deviation respectively. The value of t is the time, must be always greater than 0. The value of μ can range between -∞ and +∞, and σ is always greater than 0.

3.3.4 Gamma Probability Density Function

The PDF of two parameters Gamma distribution function can be expressed as:

$$f(t,a,b) = t^{a-1} \frac{exp^{(-\frac{t}{b})}}{b^a \Gamma(a)} \qquad (26)$$

where Γ(a) is the Gamma function evaluated; a is the shape parameter; and b is the scale parameter. The parameters in a and b must all be positive, and the values in t must lie on the interval [0, ∞).

3.3.5 Beta Probability Density Function

The PDF of two parameters Beta distribution function can be expressed as:

$$f(t,a,b) = \frac{1}{B(a,b)} t^{a-1}(1-t)^{b-1} \qquad (27)$$

where *B*(a,b) is the Beta function; the notations a and b are the shape parameters, and they are always positive numbers.

The Beta PDF provides all possible shape types based merely on the magnitude relationship between the parameters a and b.

The shape of the PDF is:

1. Positively skewed or prior – peak shape, when b > a > 1;
2. Symmetrical or midpeak shape, when a = b;
3. Negatively skewed or posterior – peak shape when a > b > 1.

3.3.6 Weibull Probability Density Function

The Weibull PDF is a continuous probability distribution that can handle increasing, decreasing or constant failure-rates and can be created for data with and without suspensions (non-failures).

The two-parameter Weibull distribution can be represented as follows:

$$f(t,a,b) = \frac{b}{a^b} t^{b-1} exp^{-(\frac{t}{a})^b} \qquad (28)$$

where

b is the slope or shape parameter; and

a is the characteristic life or scale parameter.

3.4 Calibration

In this section we will explain the theory relative to the modelling calibration, following the inverse method that was explained in the literature study.

3.4.1 Inverse calibration methodology

The inverse parameter estimation of the PDFs is formulated as a nonlinear optimization problem. The discharge model parameters are optimized by minimizing a suitable objective function based on the deviations between observed and predicted flow.

The objective function OF can be expressed as follows:

$$OF(\vec{b}) = \sum_{i=1}^{N} wi\left[O(t_i) - P\left(t_i, \vec{b}\right)\right]^2 \qquad (29)$$

where OF($\vec{b}$) is the objective function of parameter vector $\vec{b}$ that represents the error between measured and simulated values; O(t_i) and P(t_i) are observed and predicted values (hydrographs) using parameter vector $\vec{b}$, respectively; t is the time; N is the number of measurements available; and w_i is the weight of a particular measurement (Lambot et al., 2002).

The minimization of the OF will be carried out using the Global Multilevel Coordinate Search (GMCS) algorithm (Huyer, W. et Neumaier, A. 1999), described in the literature study.

In our study, we will use as stopping criterion for the GMCS optimization task the maximum number of iterations, namely α_{max}. Also is defined an additional maximum number of function evaluations β_{max}, for a specific optimization problem.

Both α_{max} and β_{max} are not known a priori for a specific optimization problem. These criteria should be defined such that not too many function values are needed after the global minimum has been reached, and such that convergence is reached. To solve this problem, we adopted the following strategy. Since local optimizations are performed many times, a too large β_{max} is likely to increase considerably the total number of function calls. Therefore, β_{max} was held to a relatively reasonable value. Nonetheless, it could happen that, using this limit, the global minimum found by GMCS is not quite optimal. Therefore, it was necessary to combine the sequentially GMCS with an additional local search algorithm with the classical Nelder-Mead Simplex algorithm (NMS) (Lagarias et al., 1998), to improve the solution.

The local search algorithm starts from the GMCS solution and stops when iterations do not improve significantly the solution. This strategy improves the efficiency of the overall optimization task since GMCS needs only to find an approximate solution (Lambot et al. 2002).

3.4.2 Calibration procedure

The calibration is carrying out in order to obtain the values of the parameters that minimize the objective function described above. These parameters values will allow representing the hydrograph with the minimum error.

In the optimization procedure, the parameter uncertainty is determined on the basis of the parameter variance-covariance matrix C (Koll and Parker, 1988).

$$C = \frac{e^T e}{n-p} H^{-1} \qquad (30)$$

where H is the Hessian matrix (size p x p) whose elements $H_{i,j}$ are defined as follows:

$$H_{i,j} = \frac{\partial \Phi(\mathrm{b})}{\partial b_i(b_j)} \qquad (31)$$

assuming that b is the estimated parameter vector of the global minimum of the objective function.

Note that when the estimated parameters are correlated, the consideration of only the diagonal terms of C may lead to an overestimation of the actual variability of the parameters, and thus of the confidence intervals (Carrera and Neuman, 1986).

The elements $A_{i,j}$ of the parameter correlation matrix A (size p x p) are defined as:

$$A_{i,j} = \frac{C_{i,j}}{\sqrt{C_{i,j}}\sqrt{C_{i,j}}} \qquad (32)$$

High correlation coefficients between estimated parameters could also lead to nonuniqueness problems.

Parameter sensitivity coefficients characterize the behaviour of the objective function at a particular point in the parameter space. In this study, sensitivity is only analyzed in the vicinity of the true parameter values.

Elements $S_{i,j}$ of the sensitivity matrix S (size n x p) are computed as follows:

$$S_{i,j} = b_j J_{i,j} \qquad (33)$$

where J is the Jacobian matrix (size n x p) whose elements $J_{i,j}$ are defined as the partial derivatives $\partial e_i/\partial b_j$. The elements of J are obtained by forward difference approximation with $\partial b_j = 0.01 \times b_j$. Elements $S_{i,j}$ represents the change of the measurement variable O relative to a change of 1% of the parameter b_j, normalized by b_j so that a comparison of sensitivities between different parameters can be made independently of their magnitudes. The matrix S gives the distribution of sensitivities both in time and space (Lambot et al., 2002).

4. Water resources issues in the Tempisque catchment

4.1 Significance of the study area

This study is performed for the Tempisque catchment in Costa Rica. During the last decades, the population of Costa Rica has increased dramatically. Concretely in the twentieth century the increase was higher than 3 million peoples (Hunt, 2009). Taking into account that the population in 2013 was 4,7 million (INEC, 2013), the increase population was enormous. It triggered an increase in the national needs for food, water and energy.

The study region is particularly important for agriculture. The region has a high potential for sugar cane and rice production. As a consequence, agricultural and hydraulic infrastructure has been strongly developed. In particular, extraction of surface and groundwater for irrigation increased substantially.

Also, multi-purpose plants haves been constructed to support the development of irrigated agriculture, to support energy production and to provide domestic water supply.

The study region is also important for tourism, due to the cultural heritage of the basin. Indeed, the Tempisque catchment is one of the richest and most beautiful catchment of the country. The tourism sector strongly developed, resulting in some megaprojects and infrastructure developments to control floods in the catchment. Also the water consumption increase for domestic purposes increased considerably.

All this have caused considerable impacts on the region's economical and natural systems. The pressure exerted by these recent developments threatens the ecological integrity of the region.

The basin presents several problems like perturbation of the hydrodynamic and intensity of the stream flows, increase in the agrochemical, nutrients, and sediment loads, uncontrolled water extraction from groundwater and surface water storages. All this involves environmental degradation and water-related social conflicts.

Given all these pressures, appropriate management strategies for the basin's water resources are badly needed. Management strategies could involve the optimization of water allocation strategies, meeting the different water demands, including the demand for preserving the rich diversity of the ecosystem. In all cases, water allocation strategies should be built on a depth understanding of the hydrological processes driving the water flow in the watershed.

This study aims unravelling the hydrological mechanisms that determine flow in the Tempisque catchment. We will use transfer function based hydrological modelling concepts to simulate the flow dynamics in some locations of the catchment in terms of rainfall and dynamics measured in other locations.

Specifically will be develop a model which will be calibrate with river discharge data in the upper Tempisque measured from 1980 to 1985.

4.2 General characteristics of the study area

4.2.1 Location, main affluent and extension

This study focuses on the 3405 km^2 Tempisque river watershed situated in the nothwestern of Costa Rica. It is the second largest basin of Costa Rica, after the Térraba river catchment. The Tempisque river watershed, including the Bebedero basin, has an extension of 5404 km^2.

The Tempisque river belongs to the Pacific Ocean basin. It is situated entirely in Costa Rica. It is the third most extensive river of the country, after the Grande de Térrava and the Reventazón rivers. It is the main river of Guanacaste province. The river flows from the Guanacaste Cordillera, near the Orosí Volcano, and discharges into the Gulf of Nicoya. It has a length of 144 km, it borders the Nicoya Peninsula and passes through Palo Verde National Park.

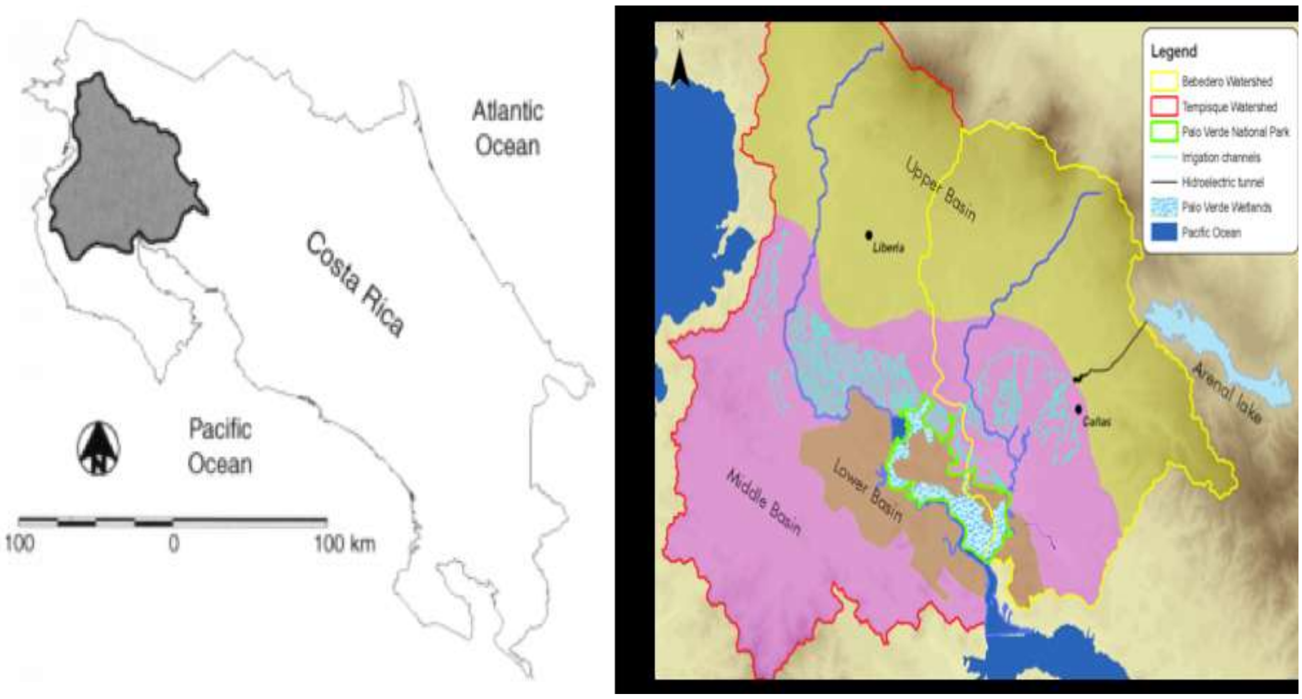


Figure 9: Map of the Tempisque-Bebedero watershed, including the Palo Verde National Park and wetlands (Alonso, 2013).

The main affluents of this stream are the rivers Bebedero, Colorado, Liberia and Salto. The river is navigable during 36 km approximately, between the river mouth and the Bolsón village.

The Tempisque basin is divided in three sections. The upper part is covered with forests, localized within the Guanacaste Volcanic Cordilllera. The middle part, is characterized by a rolling undulating landscape and agriculture land use, it is localized in Santa Rosa Plateau. The lower part is composed of plains and marshes known as “la Bajura del Tempisque”.

4.2.2 Climate

The climate in northern Pacific of Costa Rica is considered tropical. Therefore it is warm and isothermal. The pattern of precipitation is very irregular with a dry and a wet season.

The average annual temperature is 27.4°C and a mean annual precipitation of 1817 mm, although this data is highly spatially variable. The major part of the precipitation occurs during the wet season, between May and November. However there exist a short period of reduced rainfall during June, July and August known as the “Veranillo de San Juan”. This climatic phenomenon is due to an increase in the speed of the trade winds and a southerly shift of the Intertropical Convergence Zone. Droughts occur relatively frequently during the dry period and floods during the rainy season.

4.2.3 Geology, Geomorphology and Soils

The Tempisque River Basin is composed of a series of formations that correspond to different geologic ages. The Nicoya Complex is the oldest, it has the orogenesis during the Inferior Cretacic (Jimenez et al., 2001). This complex is partially overlain by intrusive rocks and the Sabana Grande and Rivas Formations that flourish in the lower areas of the basin.

The Aguacate Complex was formed during the Superior Miocene and is located on the eastern section of the watershed. The Barra Honda and Brito Formations are located in the right margin of the Tempisque River and a series of hills in the lower basin.

The Liberia and Bagaces Formations are found mainly along the Inter-American Highway. During the recent Quaternary era, lahars and volcanic structures were formed, situated in the Miravelles and Tenoria volcanoes piedmont.

Furthermore there are fluvial, alluvial and coastal deposits, which form the floodplains of the Tempisque River valley or The Tempisque Depression. The floodplains include parts of the Gulf of Nicoya.

The topography is very variable. In the upland catchment, surface height varies between 0 and 2,002 masl. In the lower catchment, there is an extensive area of wetlands.

There are 20 subgroups of soils in the Tempisque Watershed that are part of five taxonomic orders including alfisoles, entisoles, inceptisoles, mollisoles and vertisoles.

4.2.4 Flora and Fauna

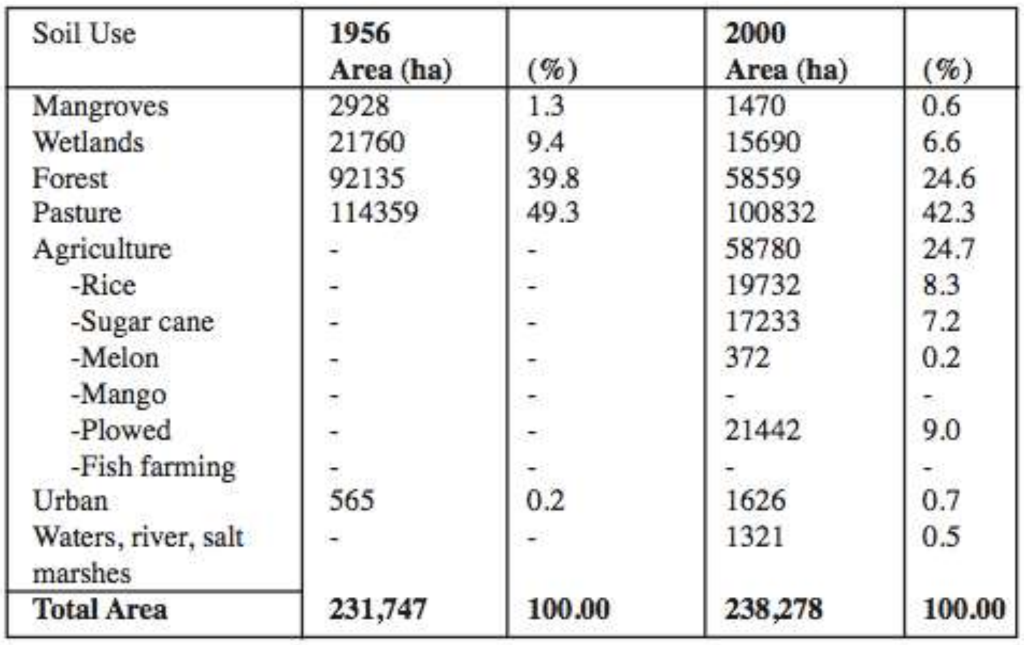
The Tempisque river watershed is one of the most beautiful landscapes of Costa Rica. It contains a big diversity of environment like cloud forest, wetlands, volcanoes and is an important habitat for various species of crocodiles, monkeys, iguanas and birds.

In total, 148 tree species have been identified (some of which are in danger of extinction), 306 species of birds, 111 mammals, 15 reptiles and 22 amphibians.

Downstream from the Arenal-Tempisque Irrigation District is the Palo Verde National Park, it is a wetland of critical importance and attracts aquatic birds that migrate southwards during the winter season. One section of the park, Laguna Foohas, houses an estimated 50.000 birds including ducks, herons, storks, egrets, grebes, ibis, jacanas and other forest birds such as macaws and small parrots (Hazell et al., 2001).

4.2.5 Land use

Fifty years ago, the Tempisque river basin land use still had approximately 50% of its original forest cover. Nowadays, most part of this area has been replaced by agricultural and livestock land use or human settlements. Practically all of the land used for agricultural purposes (90%), specially cane and rice fields, is concentrated in the lower basin (Jiménez et al., 2005), therefore in that area of the basin have been produced the major changes, specially important during the last two decades.



Soil Use	1956 Area (ha)	(%)	2000 Area (ha)	(%)
Mangroves	2928	1.3	1470	0.6
Wetlands	21760	9.4	15690	6.6
Forest	92135	39.8	58559	24.6
Pasture	114359	49.3	100832	42.3
Agriculture	-	-	58780	24.7
-Rice	-	-	19732	8.3
-Sugar cane	-	-	17233	7.2
-Melon	-	-	372	0.2
-Mango	-	-	-	-
-Plowed	-	-	21442	9.0
-Fish farming	-	-	-	-
Urban	565	0.2	1626	0.7
Waters, river, salt marshes	-	-	1321	0.5
Total Area	**231,747**	**100.00**	**238,278**	**100.00**

Table 1. Change of Land Use in the lower basin of the Tempisque River. 1955 – 2000 (Jiménez et al., 2005).

Until the 1980s, the cattle ranching was a very popular and lucrative business, but due to the decrease of the prices of products in international market, this activity diminished considerably.

Approximately 20% of the basin is occupied by wetlands. The wetland area corresponds to 1025 km^2. This environment is important for protecting numerous wildlife species, for providing water to human settlements, and for decreasing the impacts of flooding.

There are four types of wetlands in the Tempisque watershed: riverine, palustrine, lacustrine, and estuarine (Jimenez et al., 2001).

4.2.6 Economic activities

The Tempisque watershed is one of the most economically productive regions in the country. Sugar cane and rice plantations are the most productive of the country. Approximately 45% of all rice, 50% of sugar cane and all of the national melon production comes from the Tempisque river basin (Jimenez et al., 2001).

The rich cultural heritage has promoted the tourism industry, and hence prosperity and employment to the area.

The Gulf of Nicoya is one of the most productive estuary system of the world and supplies approximately a quarter of all fish production in the country as a whole (Hazell et al., 2001).

4.3 The floods

The wet season is between May and November, but normally the heaviest rainfall that produces floods, occur in the period August – October.

During flooding, water of the Tempisque inundates the riparian area and alluvial plain. The alluvial plains have been formed by numerous flooding associated with sedimentation.

Although flooding occurs regularly in the catchment, they are highly unpredictable. Flooding is not necessarily negative. These events offer a number of important benefits, such as maintaining the ecological integrity of very important ecosystems including herbaceous wetlands, where hundreds of aquatic bird species nest and many species of fish and reptiles thrive (Jimenez et al., 2001). Further floods provide a set of benefits to the farmers which grow sugar cane, rice and melon in very fertile soils.

But also floods induce damages to different cultures. For example in 1999, 2000 ha of rice were severely damaged, and 250 completely lost. In addition, 10 000 ha of sugar cane were affected and 200 devastated. As a consequence, more than 200 people lost their jobs (Jiménez et al., 2001).

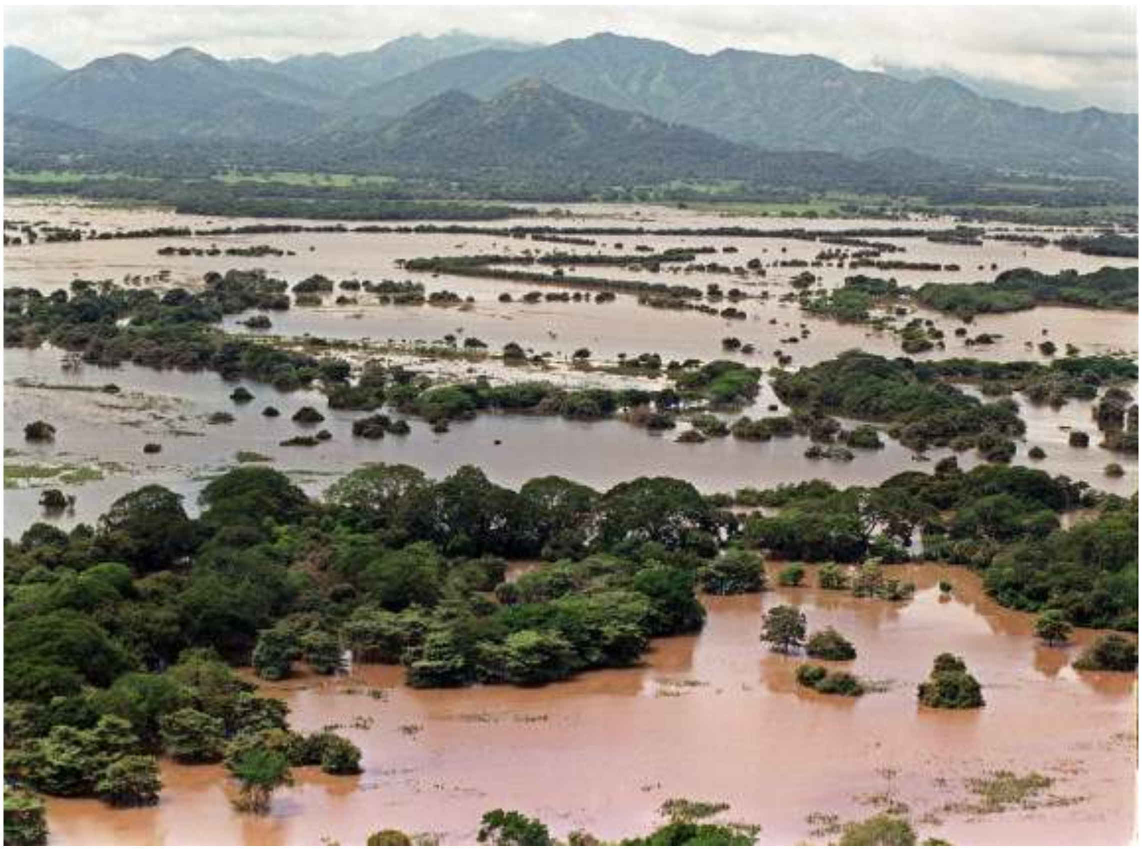


Figure 10: Floods in the Tempisque river basin during Hurricane Mitch, 1999 (Jiménez et al., 2001).

Hence, the floods produce positive and negative effects, they are necessary for the maintaining of environment but produces great damages for agriculture and settlements humans.

In the past decades the Costa Rica's Government developed a set of measure to evacuate abundant water during wet periods including engineered infrastructures, as the Senara channel, dikes and dredging. Nevertheless the floods have continued doing practically the same effects.

4.4 A brief historical perspective

Over many centuries, the Tempisque river basin was subjected to many transformations from anthropogenic origins. As a consequence, the landscape progressively changed from a dry tropical forest during the pre-Columbian period to extensive cattle ranching period during the colonial time.

Transformations in the pre-Columbian history of this basin can be discussed for four periods: the Biocrome in Zones (300 BC – 300 AD), Ancient Polychrome (300-800 AD), Mid Polychrome (800-1200 AD), and Late Polychrome (1200-1500 AD).

The main inhabitants of the region in the pre-Columbian period were the chorotega and nicarao groups that lived in semi-urban settlements, located near water sources.

After arrival of Spaniards in the region, significant changes were produced in the environment. Spaniards established different settlement patterns, implemented new types of land uses and developed different forms of spatial organization.

During the initial stages of colonial period (1502 to 1821), the subsistence agriculture was replaced by a slightly more intensive agriculture.

During this period the land was distributed in three different categories, "caballerías", "estancias" and "sitios", for use in crop production, cattle ranching and grazing, respectively (Jiménez et al. 2001). Other categories of land tenure existed such as those promoted by the Franciscan brotherhood to provide funds to support the church's activities. This land tenure consisted in farms establish around the villages and separated into small parcels where corn, cotton, cacao and plantains were produced.

Also Spaniards introduced cattle, horses, asses, pigs, goats and chickens and the region became an important exporter of beef, lard, leather and cheese to Panamá, Nicaragua, Salvador and Guatemala.

During the XIX century and the first half of the XX, the land use did not change considerably. But at the beginning of the XX century other economic activities emerged, like the exportation of precious woods.

Species such as cocobolo, cenízaro, cristobal were used for construction. Other species like cedro, caoba and guayacán were harvested for crafts, and palo brazil and mora for dye extraction. The rise of this activity produced a great decrease in the forest surface.

After 1950, the region was subjected to important transformations. The Interamerican Highway was built along with numerous other rural roads. The banking system provided credits and gave an impulse to novel economical activities and the State changed the economic politics.

The region became an exporter of beef towards North American markets. By the end of the 1970s, Costa Rica was the fourth main exporter of meat to the USA.

In North America the import activity of beef was a highly unstable business. During the1980s beef demand stagnated in the USA. The industry of the region decreased and furthermore Costa Rica suffered an economic crisis that accelerated inflation and skyrocketed interest rates.

4.5 The Arenal - Tempisque Irrigation Program

This Arenal - Tempisque Irrigation Program (known as PRAT in its Spanish acronym), was conceived originally for a twofold objective, by one hand provide hydroelectricity production through a hydroelectric plant, and by the other hand the irrigation purposes. In the figure 11 is shown the irrigation canals and the hydroelectric plant.

The original irrigation programme was conceived for the irrigation of an area upper than 60 000 ha. The catchment is divided into two districts, the Arenal and Zapandi district. This project jurisdictionally affects the cantons of Abangares, Cañas, Bagaces and Liberia, although it influences also the cantons of Tilarán, San Carlos and Nicoya.

The program was developed between 1975 and 1978, and is managed by The National Service of Subterranean Waters, Irrigation and Drainage (SENARA).

The irrigation project produced changed considerably in the landscape of the region, including those areas that are affected by the project downstream of the irrigation perimeter, such as wetlands and the coastal and marine area of the Gulf of Nicoya.

The water requirements for the irrigation project are met by means of the Arenal reservoir, situated northwest of the basin.

The irrigation project exploits the water from the Arenal reservoir for agriculture and domestic water supply, through a great channel network in the lower basin. The nominal discharge of the channel is approximately 60-65m^3.sec^{-1} during the dry season and between 30-35m^3.sec^{-1} during the rainy season.

In total, more than 255 km of canals and 252 km of roads and paths have been built, to supply water for the irrigation area that it is between 40 000 and 44 000 ha (Jiménez et al. 2001). The irrigated area is mainly covered by two crops, rice and sugar cane, with respectively 50 and 40% of the crop area (Hazell et al., 2001).

Water from the Arenal catchment originally flowed down towards the Caribbean Sea. With this project, major water transfer took place towards the Pacific catchment, involving the development of the Arenal-Tempisque hydroelectric complex, a cascade of three electric plants (Arenal, Corobici, and Sandillal). This hydroelectric complex is one of the most important hydroelectric centrals of the National Electric Sistem. It is the third plant in the country for producing electricity, with an installed capacity of 157.4 MW (Hazell et al., 2001). Currently, it generates 12% of the total electricity production of the country (Jiménez et al. 2001; Coto, 2001).

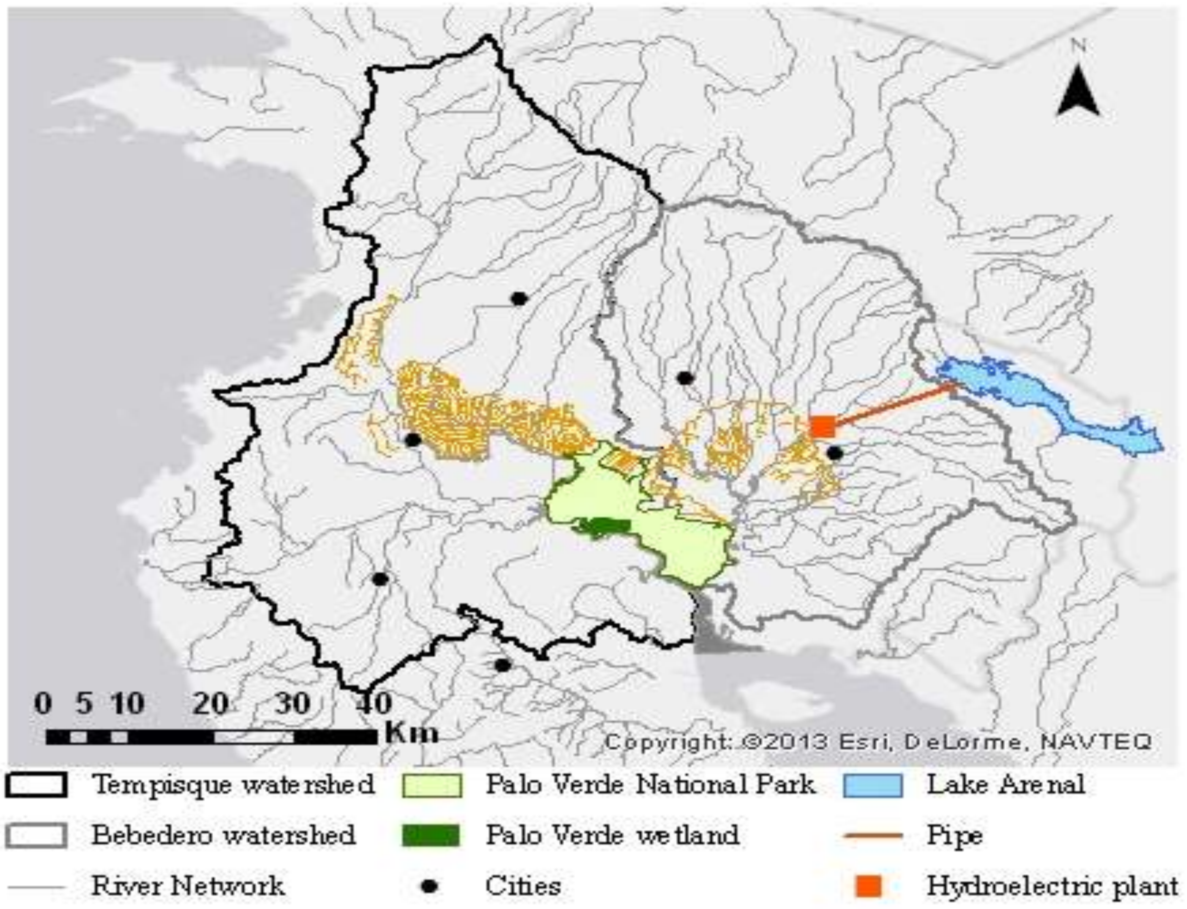


Figure 11: The Arenal – Tempisque Irrigation Program.

5. Material and Methods

In this section we will explain the initial material that we have (hydrological data), and the methods that we have followed to carry out the hydrological modelling.

5.1 Selection of data

For our purpose, we have used daily data of rainfall and discharge of upper Tempisque Basin, measured from 1980 to 1985.

There exist a large amount of rainfall historical data, gathered from the national electricity institute (Instituto Costarricense de Electricidad, ICE), and the national weather institute (Instituto Meteorologico National IMN), digitized and uploaded on the Hydrobase server (http://abe.ufl.edu/carpena/research/Tempisque.html) of University of Florida.

We have used eight stations of rainfall, and two stations of discharge, and we have applied the Transfer Functions in all combinations rainfall-discharge, in order to gain more accuracy in the interpretation of the basin behaviour. Therefore we have 16 combinations rainfall-discharge. The stations locations in the watershed are shown in the figure below.

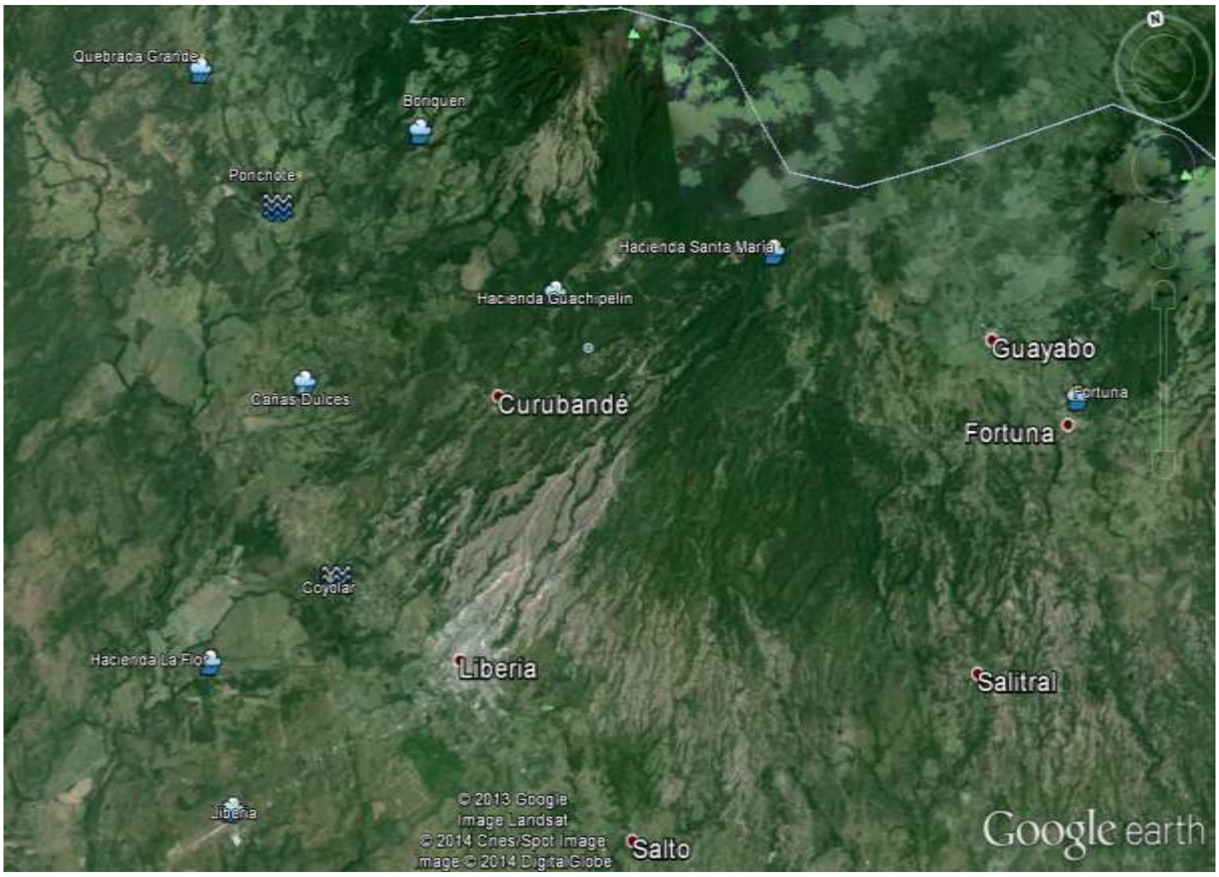


Figure 12: the stations locations in the Tempisque river watershed.

The main characteristics of the rainfall stations are shown in the next table (the elevations of the stations are given in local reference, Costa Rica uses datum WGS84):

	Hydrobase Code	The owner station	Latitude	Longitude	Elevation	Count (days)
Liberia	TBRN1	IMN	10.6	-85.5333	85 m	2071
Hacienda la Flor	TBRN2	IMN	10.65	-85.5333	50 m	2071
Cañas Dulces	TBRN4	IMN	10.7333	-85.4833	100 m	2071
Hacienda Guachipelín	TBRN5	ICE	10.75	-85.3833	520 m	2071
Boriquen	TBRN6	ICE	10.8167	-85.4167	580 m	2071
Quebrada Grande	TBRN7	IMN	10.85	-85.5	366 m	2071
Hacienda Santa María	TBRN8	IMN	10.75	-85.3	825 m	2071
Fortuna	TBRN10	ICE	10.6833	-85.2	430 m	2071

Table 2: the main characteristics of the rainfall stations.

The graphs showing the rainfall evolution in the modelling period (01/05/1980 – 31/12/1985) are presented as follows:

Liberia:

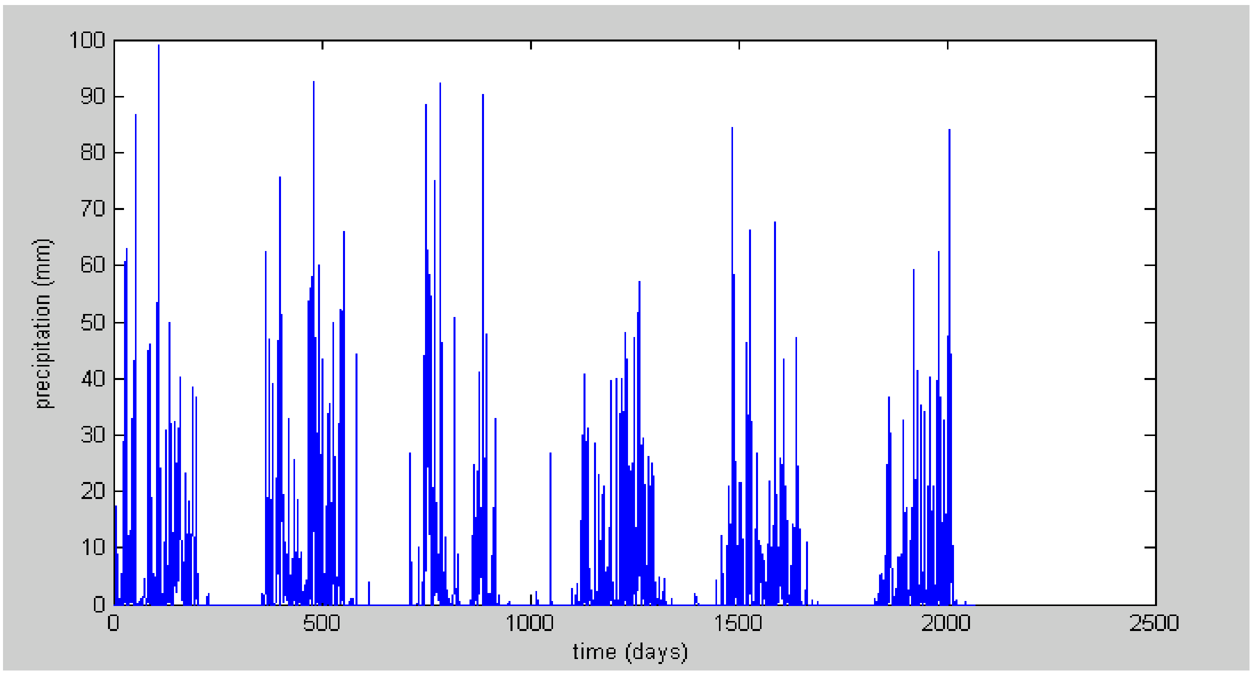


Figure 13: Graphs showing the precipitation evolution between 1/05/1980 and 31/12/1985 in Liberia.

Hacienda La Flor:

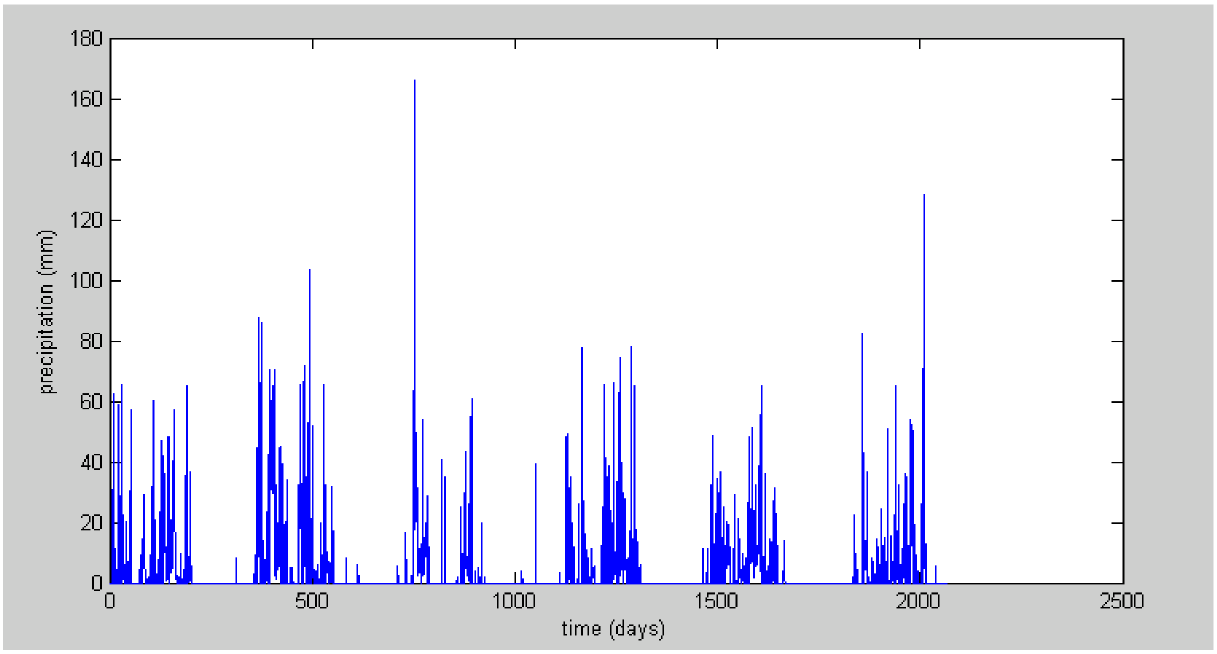


Figure 14: Graphs showing the precipitation evolution between 1/05/1980 and 31/12/1985 in Hacienda La Flor.

Cañas Dulces:

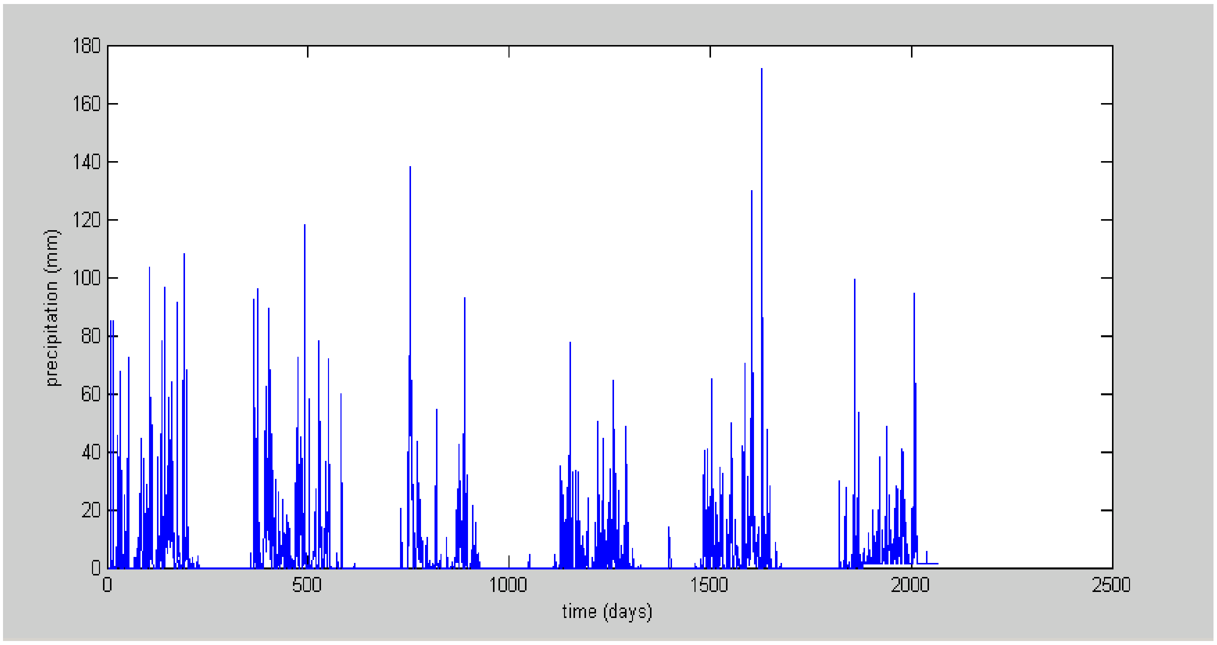


Figure 15: Graphs showing the precipitation evolution between 1/05/1980 and 31/12/1985 in Cañas Dulces.

Hacienda Guachipelín:

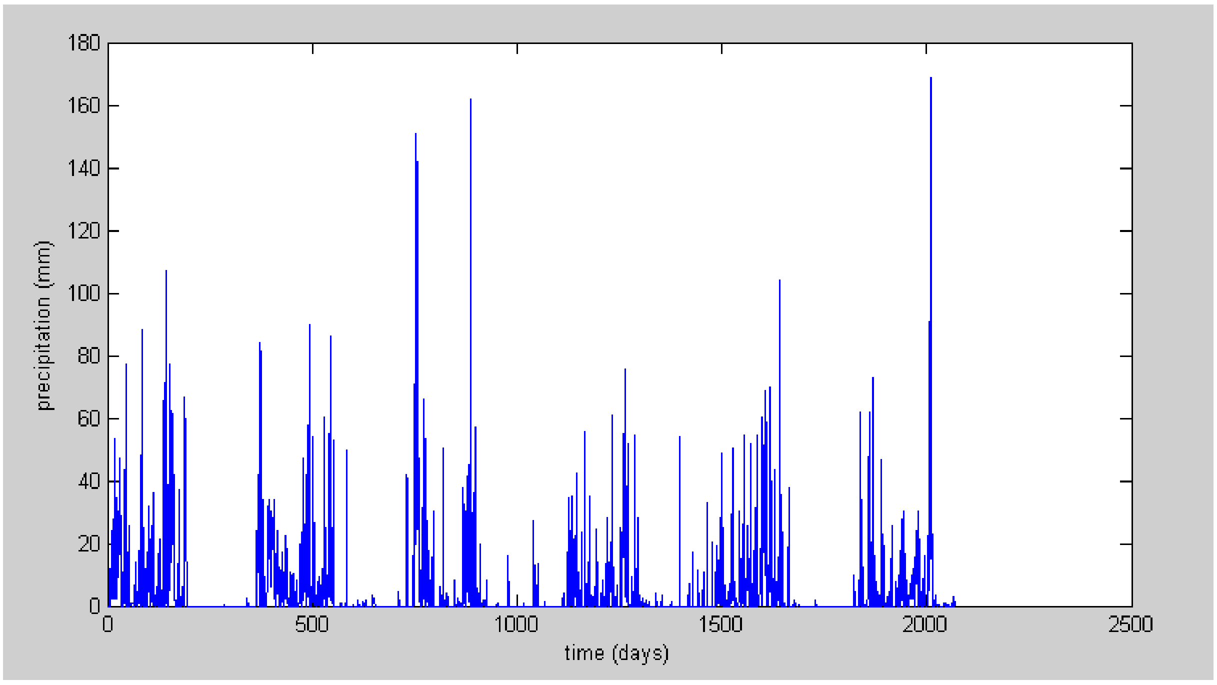


Figure 16: Graphs showing the precipitation evolution between 1/05/1980 and 31/12/1985 in Hacienda Guachipelín.

Boriquen:

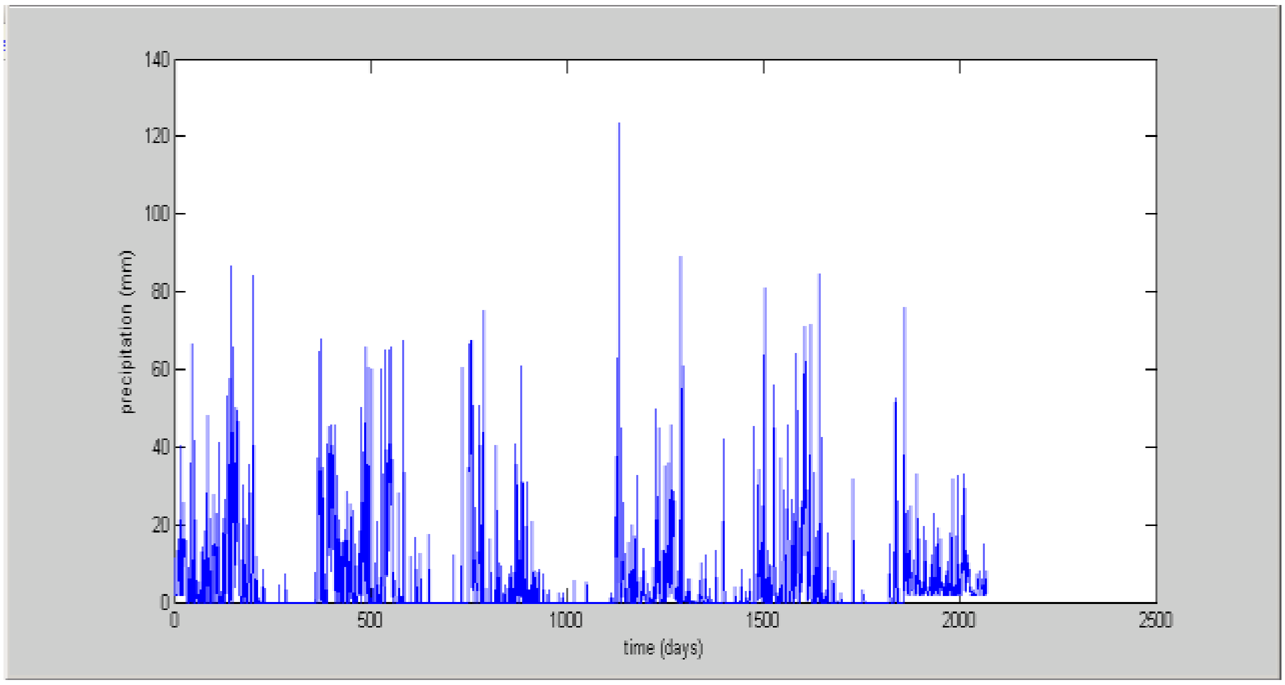


Figure 17: Graphs showing the precipitation evolution between 1/05/1980 and 31/12/1985 in Boriquen.

Quebrada Grande:

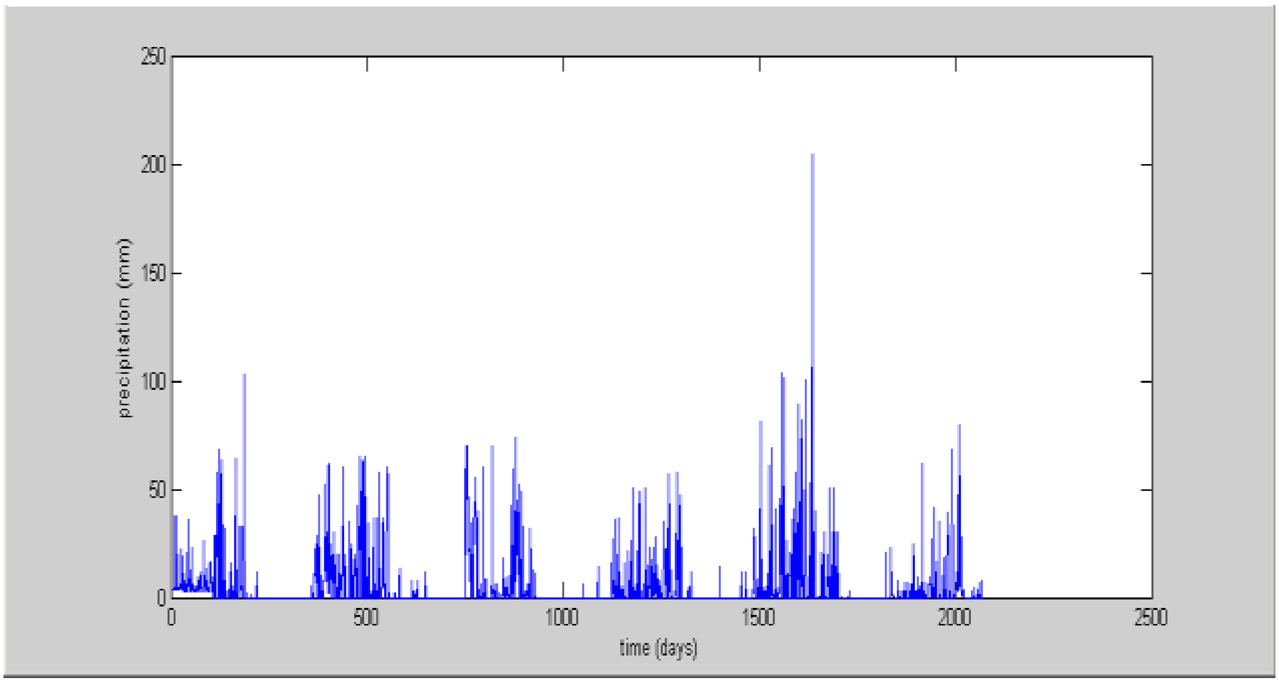


Figure 18: Graphs showing the precipitation evolution between 1/05/1980 and 31/12/1985 in Quebrada Grande.

Hacienda Santa María:

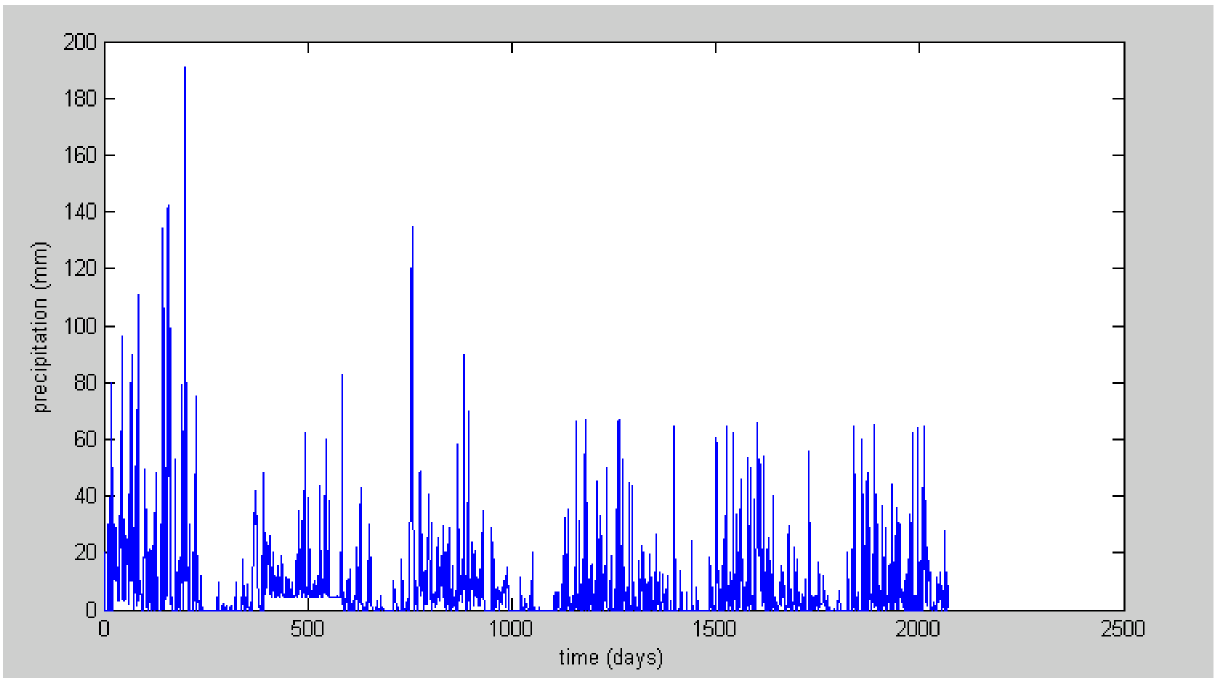


Figure 19: Graphs showing the precipitation evolution between 1/05/1980 and 31/12/1985 in Hacienda Santa María.

Fortuna:

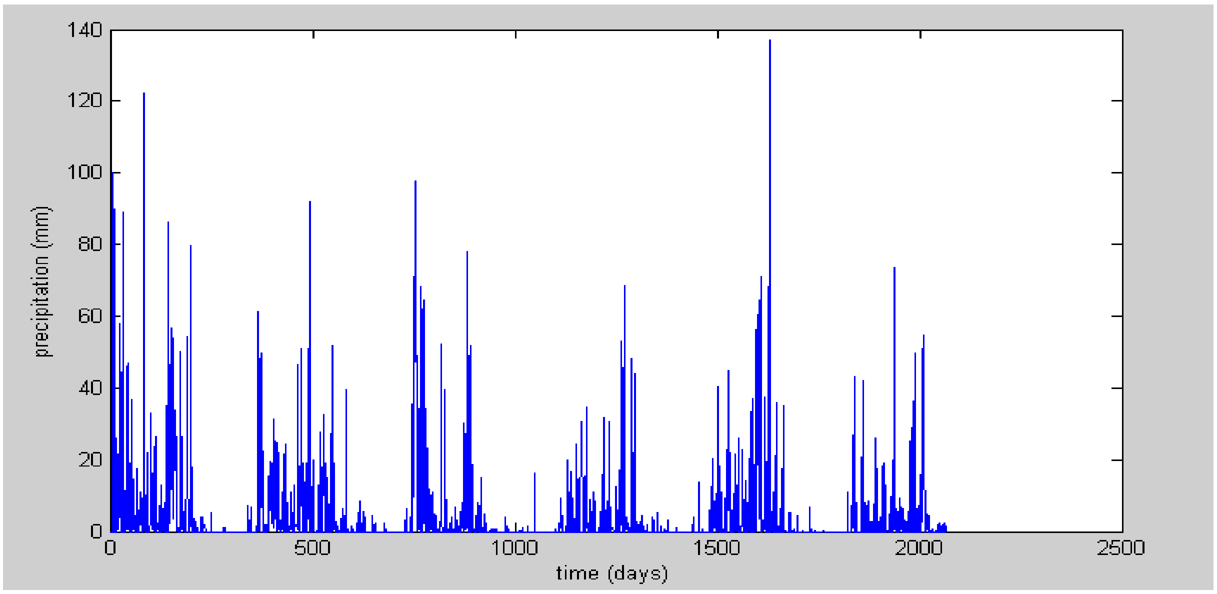


Figure 20: Graphs showing the precipitation evolution between 1/05/1980 and 31/12/1985 in Fortuna.

The discharges data were obtained exclusively from the limnigraph readings as made available by the national electricity institute (ICE). We have utilized data from two stations: Coyolar and Ponchote.

Coyolar:

The station gauge is located in Colorado River, approximately 1500 meters downstream of the Interamerican Highway Bridge. Concretely the Coyolar station is located in the Colorado River, and belongs to ICE. The station is equipped with limnigraph, pressure type (home Neyrpic).

- Hydrobase Code: TBCA2
- The owner station: ICE
- Draining area: 128,2 km^2
- Latitude: 10.6689
- Longitude: -85.4844
- Elevation (WGS84): 76.69 m
- Initial Date: 1/05/1980
- Final Date: 31/12/1985
- Count: 2071
- Frequency: daily

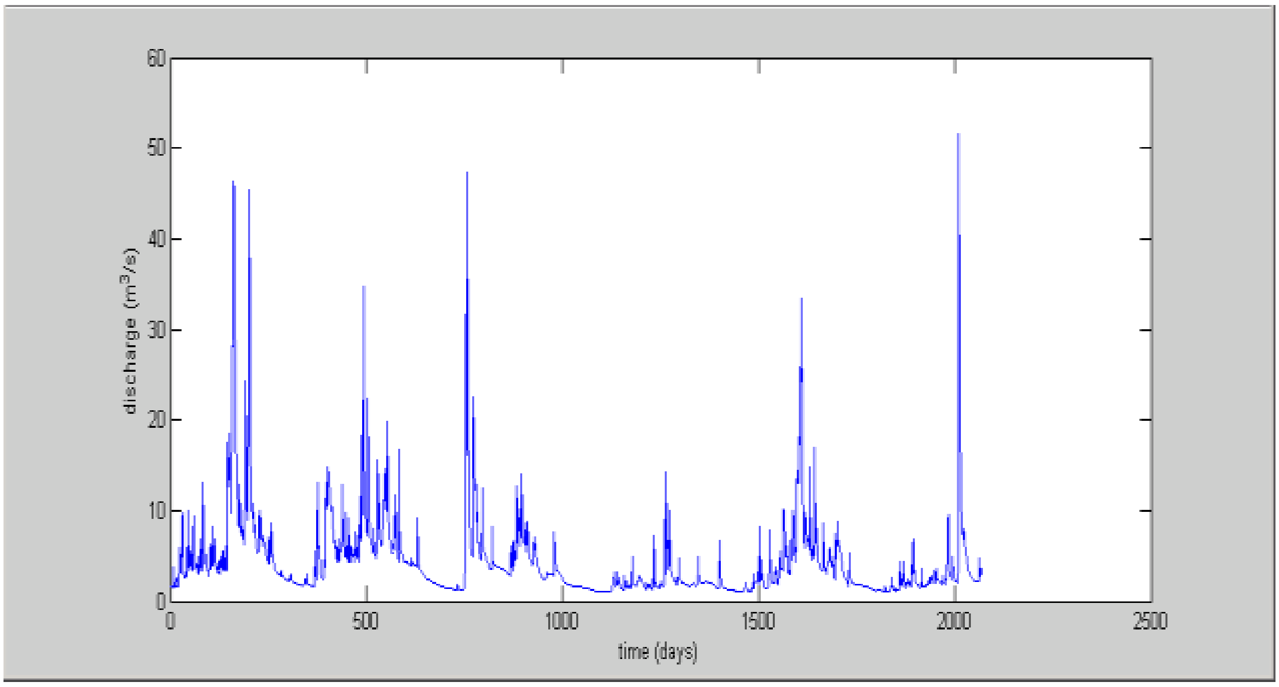


Figure 21: Graphs showing the discharge evolution between 1/05/1980 and 31/12/1985 in Coyolar.

El Ponchote:

The Ponchote station is located in Salitral River, between the Santa Anita and Ahogados villages. It belongs to ICE.

- Hydrobase Code: TBCA3
- The owner station: ICE
- Draining area: 26.2 km^2
- Latitude: 10.8047
- Longitude: -85.4711
- Elevation (WGS84): 258 m
- Initial Date: 1/05/1980
- Final Date: 31/12/1985
- Count: 2071
- Frequency: daily

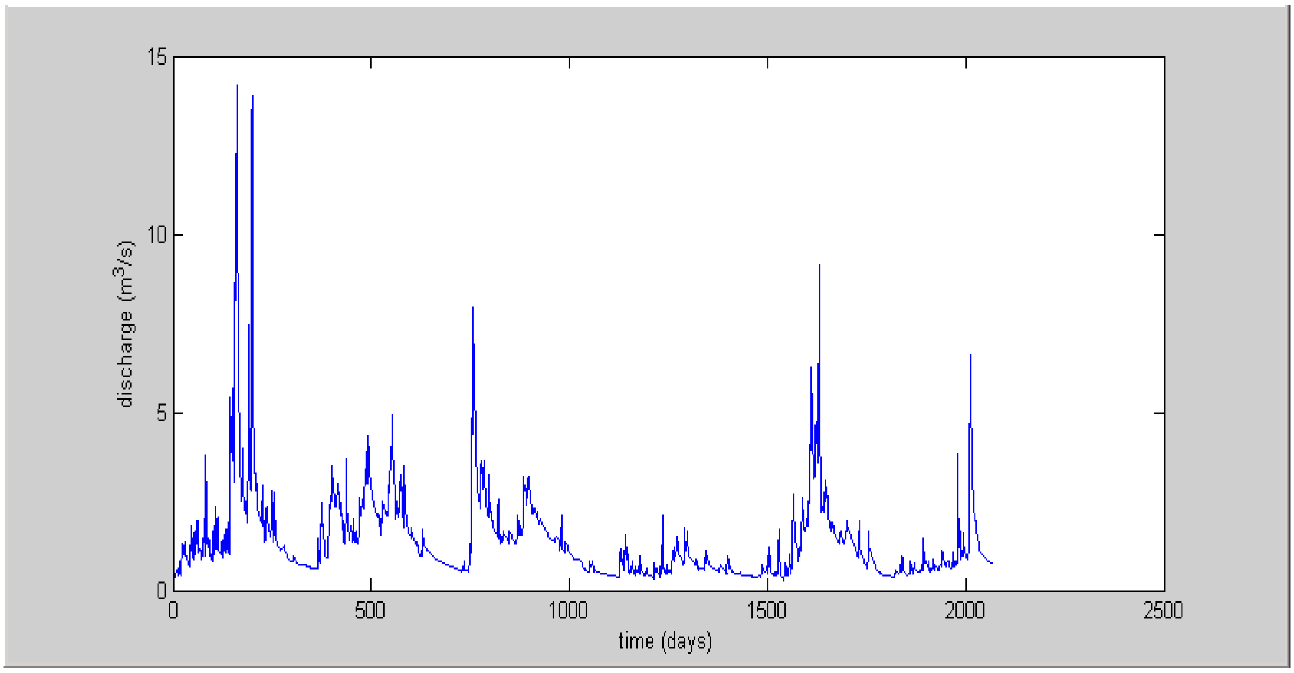


Figure 22: Graphs showing the discharge evolution between 1/05/1980 and 31/12/1985 in El Ponchote.

5.2 Implementation

In this section we will develop the procedure that we have followed to obtain the unit hydrographs, and also the established criteria to choose the best model.

5.2.1 Matlab

The implementation and calibration carried on in this study were programmed entirely in MATLAB routines (version 7.12.0.635 (R2001a)).

The rainfall and discharge data of the corresponding period were exported to programme.

All the transfer functions that we have used were codes in MATLAB routines. Detailed codes are given in the annex of the document.

The discharge function is modelled using a transfer function with their respective parameters and the precipitation, this general function can be expressed as:

$$Q_{mod} = f\ (TF(A,B),\ P) \qquad (34)$$

where Q_{mod} is the modelled discharge; (TF(A,B)) is the corresponding transfer function with the parameters A and B; and P is the precipitation.

The last expression is the mathematical convolution between the transfer function and the rainfall. The next step was developing the objective function (OF). It is obtained as the difference between the measured and modelled discharges.

5.2.2 Calibration

The Global Multilevel Coordinate Search combined with Nelder-Mead Simplex Algorithm is used in this study and implemented in MATLAB.

The first stage in the GMCS is the definition of the fixed and fitted parameters and also fitted parameters space in which the optimization process will be carried out or for which the objective function will be calculated.

To establish the lower and upper bounds (U.V) in which the optimization is carried out, we have done a small study of the behavior of every probability density function. We have selected the range where the representation of the PDF is the shape of a Gauss bell, with maximum and minimum values of width.

In the local optimization developed in Matlab, the Nelder-Mead function employed was “fminsearch”. It finds the minimum of a scalar function of several variables, starting at an initial estimate. This is generally referred to as unconstrained nonlinear optimization. The associated command to “fminsearch” is the follows:

This function starts at the point x0 (global minimum by GMCS) and returns a value x that is a local minimizer of the function described in fun, in our case it is the objective function.

Therefore we have obtained two possible solutions of parameter of each transfer function. However, the global solution with GMCS is the better, because it’s inside of the initial parameter range (U, V).

5.2.3 Criteria for model selection

The suitability of the PDFs to represent the shape of the hydrographs was obtained by comparing the simulated hydrographs with the observed hydrographs. A criterion for the selection of the PDFs is the resemblance of the shape of the PDF to the shape of the observed flood hydrograph.

In order to evaluate the goodness of fit of predicted data, we have employed three criteria, concretely The Root Mean Square error (RMSE), the Nash–Sutcliffe model efficiency coefficient (NSE), and the Error Sum of Square (SSQ).

The RMSE is a useful single measure of the prediction capability of a model, since it indicates the precision with which the model estimates the value of the depended variable. The RMSE can be defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}} \qquad (35)$$

where O_i and P_i are the observed and predicted discharge respectively; and n is the number of observations.

The NSE is traditionally used to assess the predictive power of hydrological models. It is defined as:

$$NSE = 1 - \frac{\sum_{i=1}^{n}(O_i - P_i)}{\sum_{i=1}^{n}(O_i - \bar{O})} \qquad (36)$$

where O_i and P_i are the observed and predicted discharge respectively; and $\bar{O}$ is the average of observed data.

The Error Sum of Square is a measure of the discrepancy between the data and an estimation model. It can be expressed as:

$$SSQ = \sum_{i=1}^{n}(O_i - P_i)^2 \qquad (37)$$

where n is the number of observations; and O_i and P_i are the observed and predicted discharge respectively.

6. Results and Discussions

In this section we present the results of the modelling of the rainfall-discharge relationship in some locations of the Tempisque river watershed using Transfer Function modelling. First we used data from 2 rainfall and 2 discharge stations. We have employed the precipitation – discharge series Boriquen rain – Ponchote flow, and Quebrada Grande rain – Coyolar flow.

We first identified the parametric equation that is most appropriate for modelling flow in the catchment. We tested 6 parametric models (simple TF, normal TF, log-normal TF, beta TF, gamma TF and Weibull TF).

We used visual comparison of modelled and simulated flow and statistical model error statistics to identify the most appropriate model.

6.1 Identification of the parametric TF model

In order to do the identification of the parameters of every Transfer Function, we have used only the slow TF, that is, only one Transfer Function in all rainfall range. The obtained results in the first combination (Boriquen rain – Ponchote flow) were the following:

	Distribution	**Parameters**	**RMSE**	**NSE**	**SSQ**
1	Simple T.F.	A = 2.24; B =77.6	32.2353	-6.6789	4084.6031
2	Normal	μ =14 ; σ=7	0.87910	0.52531	1600.2862
3	Log-normal	μ = 3.6 ; σ= 0.65	0.94868	0.3472	1863.8720
4	Beta	A =43.49 ; B =48.5	1.89490	-1.2056	7436.5343
5	Gamma	A =6 ; B =7	0.98882	0.36578	2045.0535
6	Weibull	A =73 ; B =6	0.97264	0.41893	1959.2198

Table 3: Parameters of the distribution and estimated values of error criteria (Boriquen rain – Ponchote flow).

Unfortunately, the peak flows were not correctly simulated with the first model. Therefore, in order to improve the simulation for the peak flow event, we implemented a model consisting of 2 Transfer Functions coupled in parallel. The first TF, the slow TF, will only be activated for small and moderate rainfall events. The second TF, the fast TF will be activated for intensive rainfall events. A threshold rainfall determines when either the slow or the fast TF will be activated.

Below we show the graphs using the two Transfer Functions coupled in parallel. These graphs display the simulated and observed discharge using the six transfer functions, in blue color are shown the observed discharge and in green color the predicted. Firstly is show the serie Boriquen rain-Ponchote flow, and after Quebrada Grande rain-Coyolar flow.

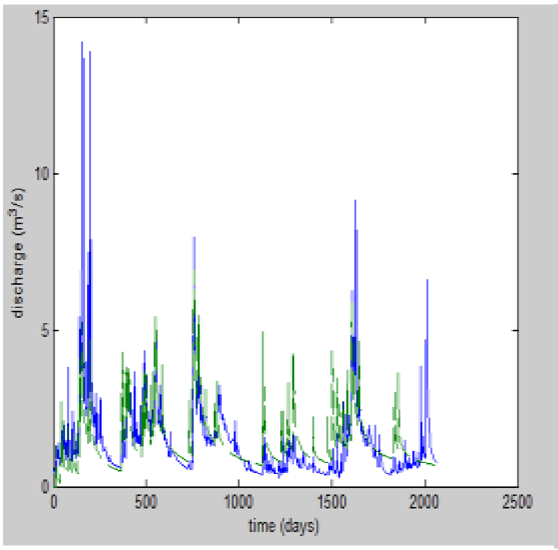


Fig. 23: Boriquen – Ponchote Weibull TF

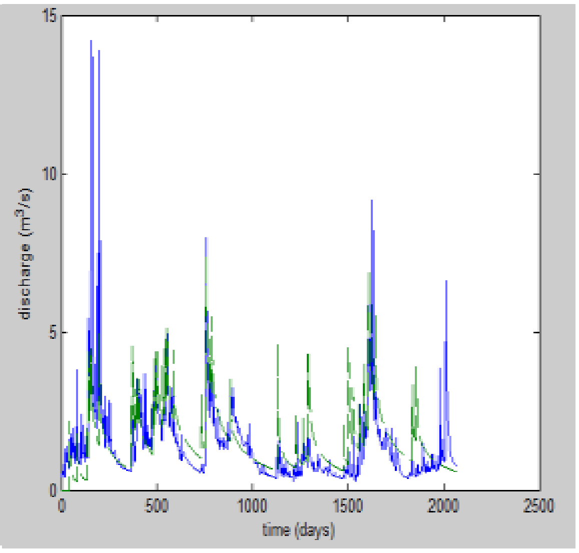


Fig. 24: Boriquen - Ponchote Gamma TF

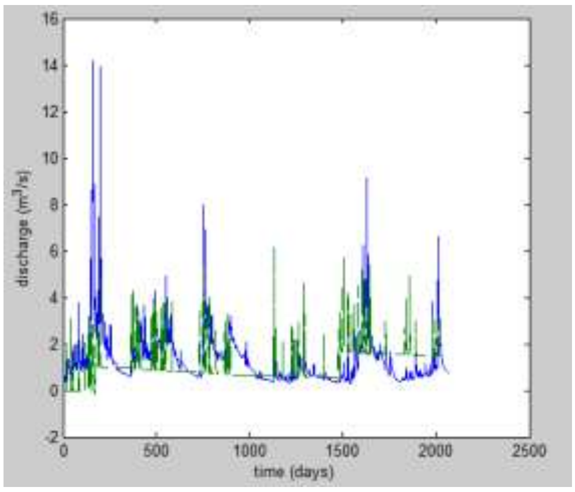


Fig. 25: Boriquen - Ponchote Simple TF

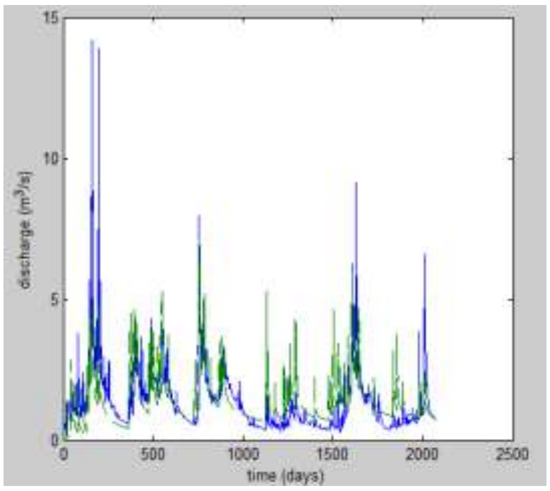


Fig. 26: Boriquen - Ponchote Beta TF

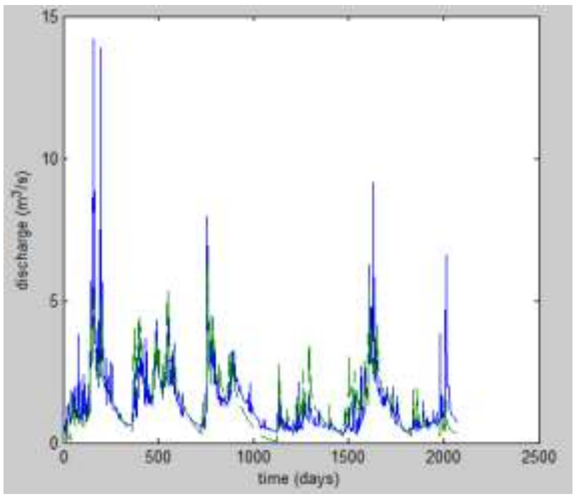


Fig. 27: Boriquen-Ponchote Normal TF

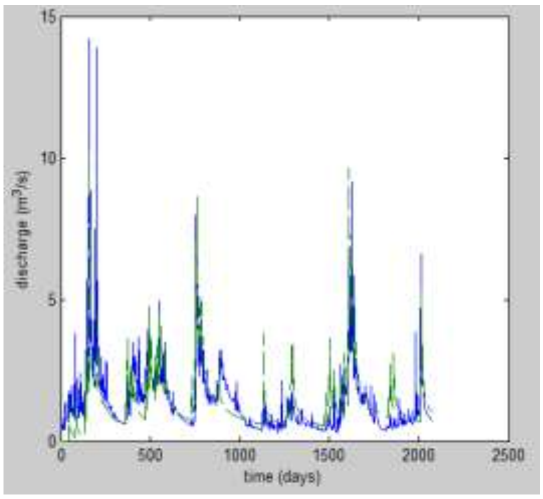


Fig. 28: Boriquen-Ponchote Log-normal TF

In the experiment Quebrada Grande rain and Coyolar flow, the obtained results were the follow (only in slow Transfer Function):

	Distribution	**Parameters**	**RMSE**	**NSE**	**SSQ**
1	Simple T.F.	A =1.6 ; B =80.2	5.330	-0.20906	58944.990
2	Normal	μ = 14; σ=7	4.454	0.19726	41085.2521
3	Log-normal	μ 3.6= ; σ=0.65	3.4715	0.18807	24957.4773
4	Beta	A =44.87 ; B =49.2	6.5723	-0.83491	84456.474
5	Gamma	A =4.0 ; B =3.2	4.3804	0.18491	39737.790
6	Weibull	A = 35.6; B =6.0	4.3831	0.18390	39784.642

Table 4: Parameters of the distribution and estimated values of error criteria (Quebrada Grande rain – Coyolar flow).

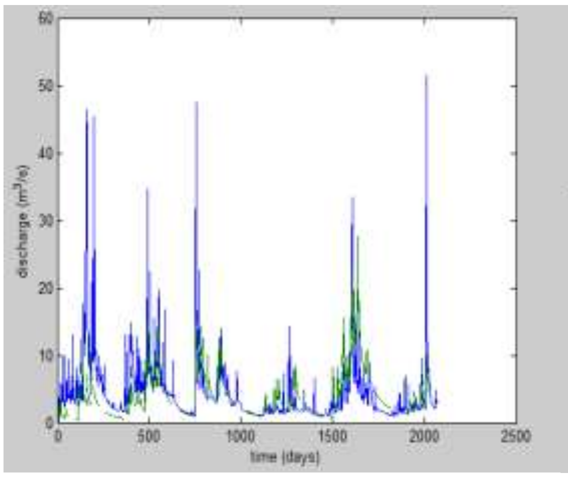


Fig. 29: Quebrada Grande – Coyolar Weibull TF

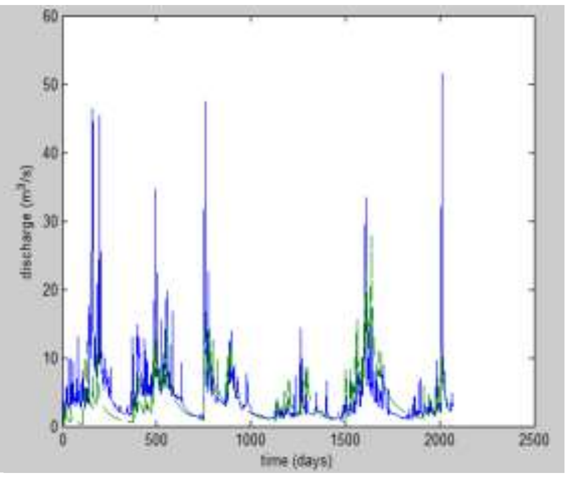


Fig. 30: Quebrada Grande-Coyolar Gamma TF

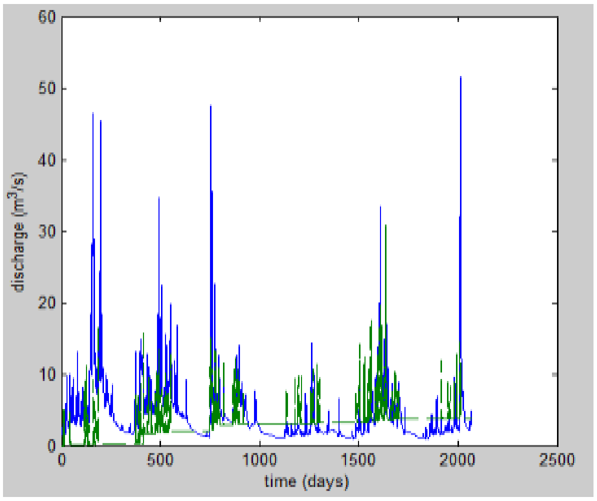


Fig. 31: Quebrada Grande – Coyolar Simple TF

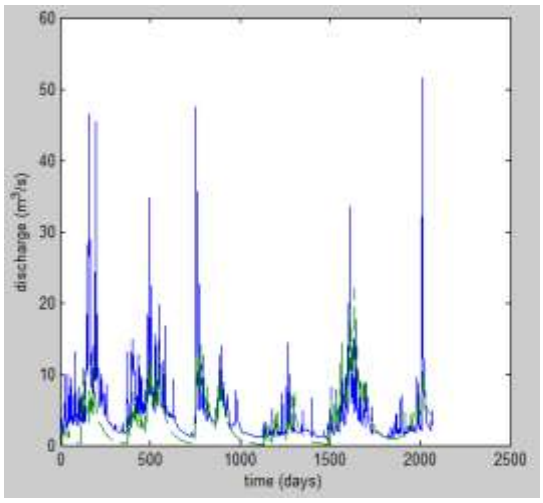


Fig. 32: Quebrada Grande – Coyolar Beta TF

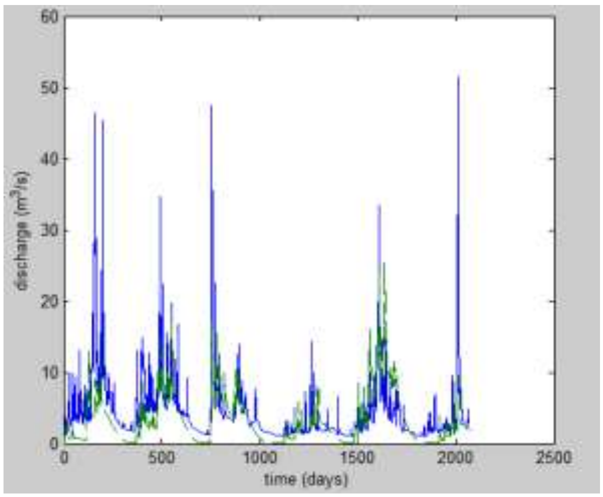


Fig. 33: Quebrada Grande-Coyolar Normal TF

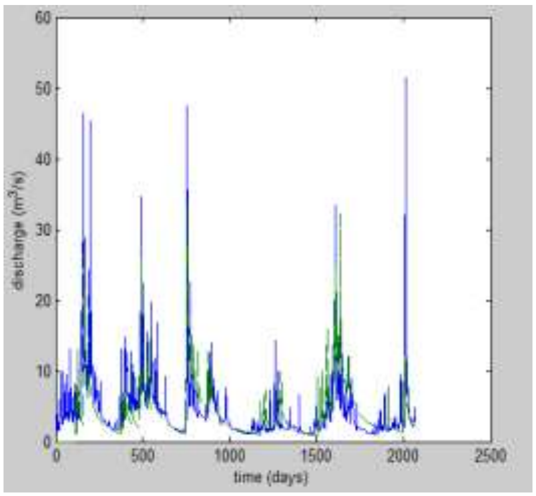


Fig. 34: Quebrada Grande-Coyolar Log-normal TF

After analysing the tables with the obtained results and the graphs, we see that the simulations done by means of the Log-normal transfer function yield the best results. However also the Gamma and Weibull transfer functions gave good results. Therefore we will implement the Log-normal transfer function in all locations of the catchment, in order to know what are the water travel time and the shape of the several unit hydrograph in the catchment.

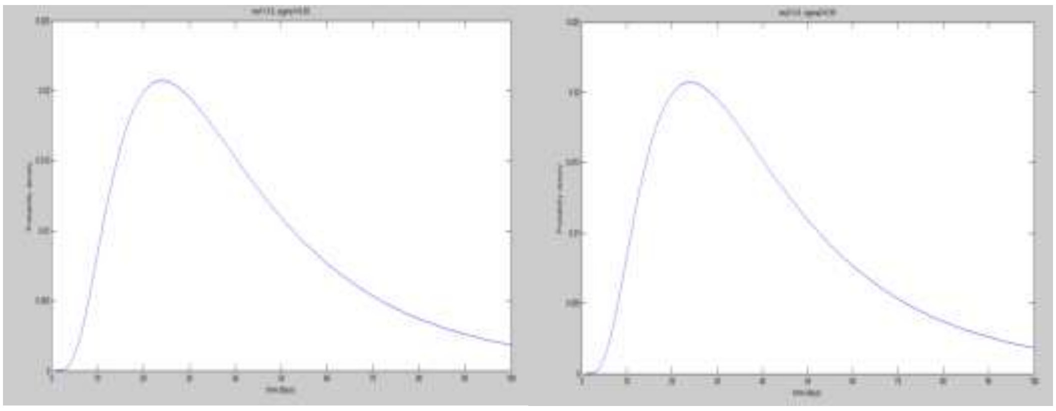
The next table show the parameters value (Mu and Sigma) using the Log-normal transfer function in all combinations rainfall-discharge in the catchment.

	PONCHOTE FLOW	**COYOLAR FLOW**
QUEBRADA GRANDE RAIN	μ_1= 3.6; σ_1=0.65; μ_2=3.6; σ_2=0.65	μ_1=3.6 ; σ_1=0.65; μ_2=1.7005; σ_2=0.0336
BORIQUEN RAIN	μ_1= 3.6; σ_1=0.65; μ_2=3.6 ; σ_2=0.65	μ_1=3.6 ; σ_1=0.65; μ_2=1.5873 ; σ_2=0.65
CAÑAS DULCES RAIN	μ_1= 3.6; σ_1=0.65; μ_2=3.6; σ_2=0.65	μ_1= 1.7832; σ_1=0.65; μ_2=3.6; σ_2=0.65
FORTUNA RAIN	μ_1= 3.6; σ_1=0.65; μ_2=3.6; σ_2=0.65	μ_1= 1.7451; σ_1=0.65; μ_2=3.6; σ_2=0.65
GUACHIPELÍN RAIN	μ_1= 3.6; σ_1=0.65; μ_2=3.6; σ_2=0.65	μ_1= 3.6; σ_1=0.65; μ_2=1.5; σ_2=0.046
HACIENDA LA FLOR RAIN	μ_1= 3.6; σ_1=0.65; μ_2=3.6; σ_2=0.65	μ_1= 3.6; σ_1=0.65; μ_2=1.5; σ_2=0.0438
HACIENDA SANTA MARÍA RAIN	μ_1= 3.6; σ_1=0.65; μ_2=3.6; σ_2=0.65	μ_1= 3.6; σ_1=0.65; μ_2=1.6983; σ_2=0.0344
LIBERIA RAIN	μ_1= 3.6; σ_1=0.65; μ_2=3.6; σ_2=0.65	μ_1= 3.6; σ_1=0.6254; μ_2=2.013; σ_2=0.025

Table 5: the parameters value in all combinations rainfall-discharge.

After the parameters identification, is possible represent the shape of the Probability Density Functions (PDFs) of the Log-normal distribution. As we have explained in the theory section, the Probability Density Function corresponds to the shape of the unit hydrograph. The slow and fast TFs identified for the different rainfall-runoff combinations in the catchment are given below.

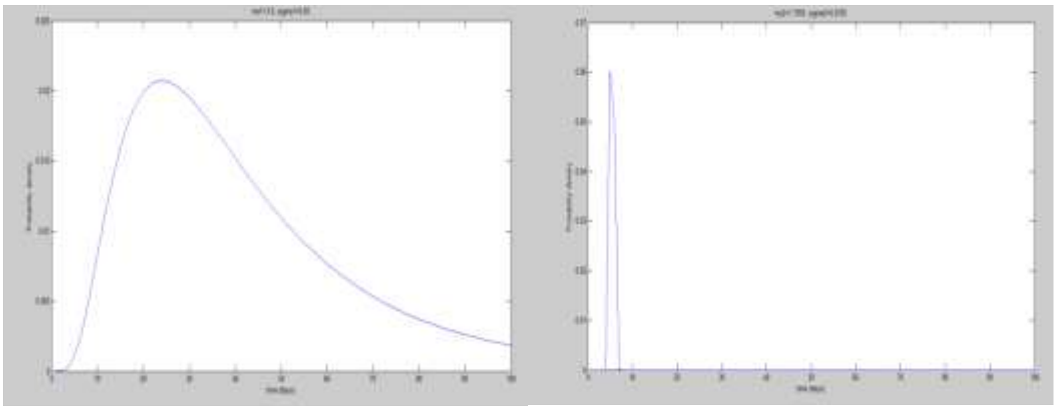
Quebrada Grande rain – Ponchote flow:



(a) (b)

Fig. 35: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

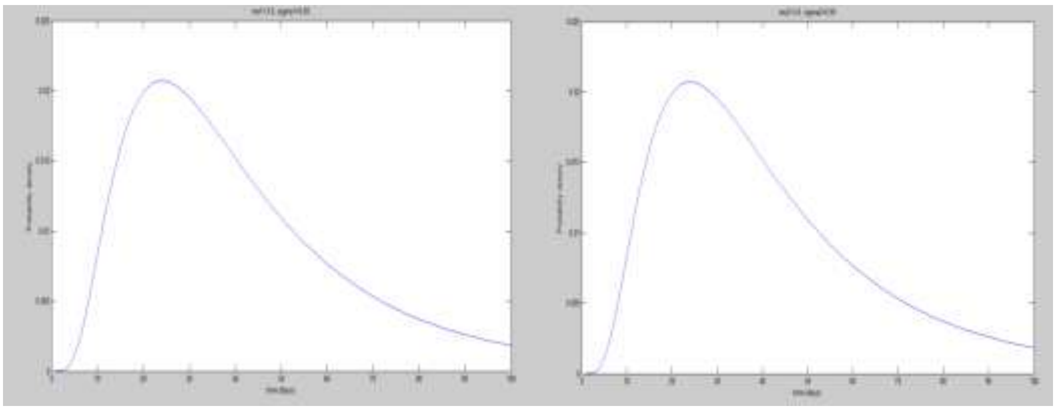
Quebrada Grande rain – Coyolar flow:



(a) (b)

Figure 36: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

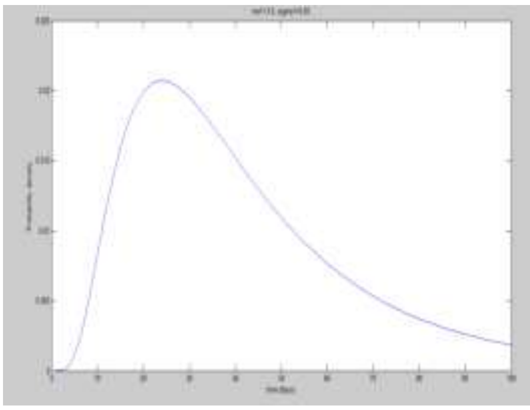
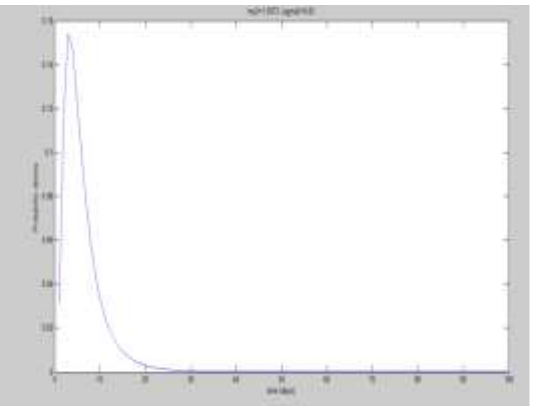
Boriquen rain – Ponchote flow:



(a) (b)

Fig. 37: optimal log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

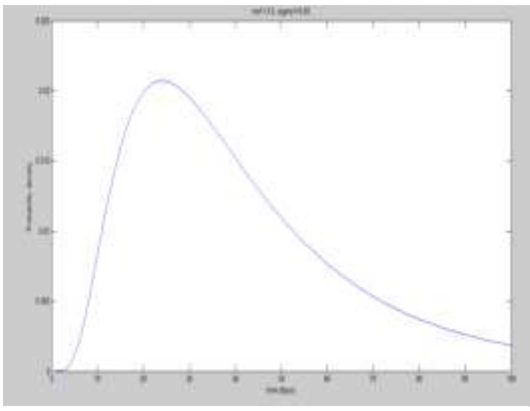
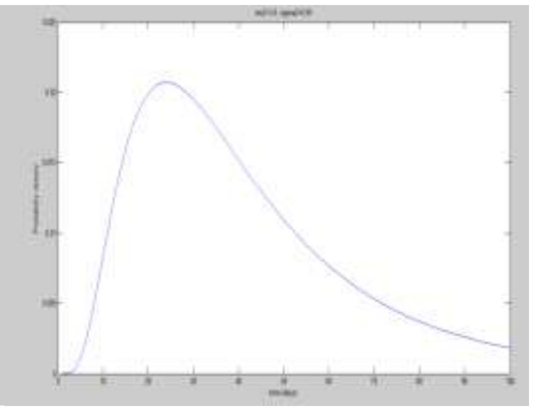
Boriquen rain – Coyolar flow:

(a) (b)

Fig. 38: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

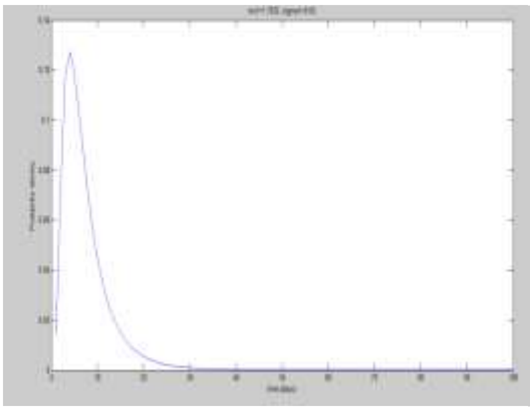
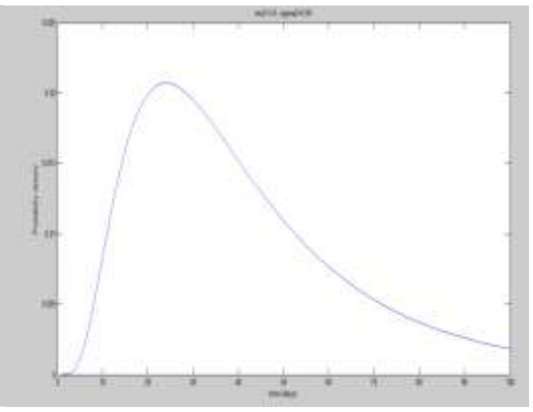
Cañas Dulces rain – Ponchote flow:

(a) (b)

Fig. 39: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

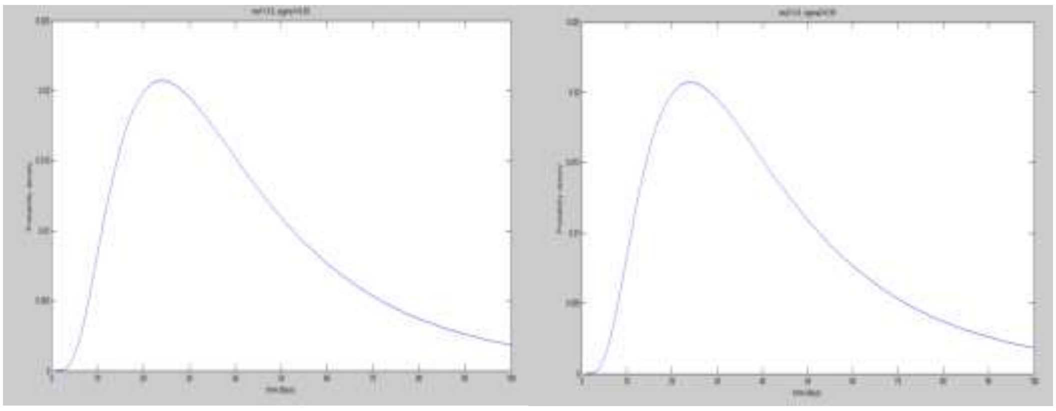
Cañas Dulces rain – Coyolar flow:

(a) (b)

Fig. 40: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

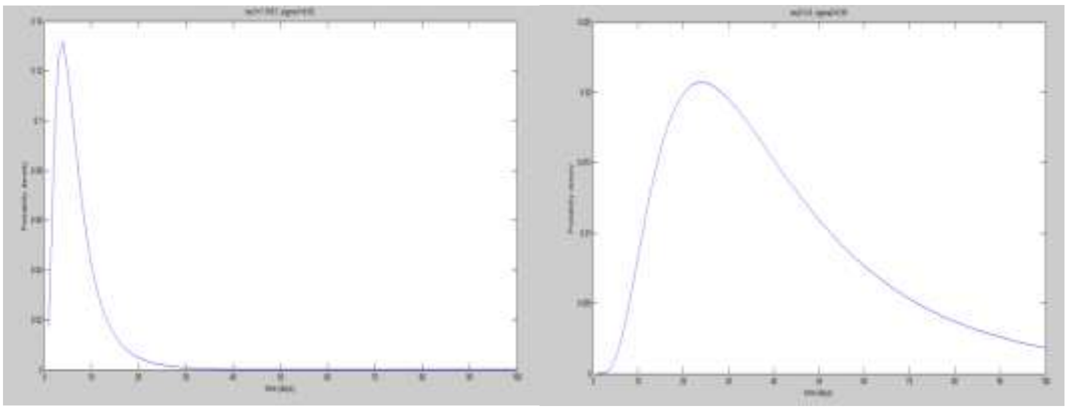
Fortuna rain – Ponchote flow:



(a) (b)

Fig. 41: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

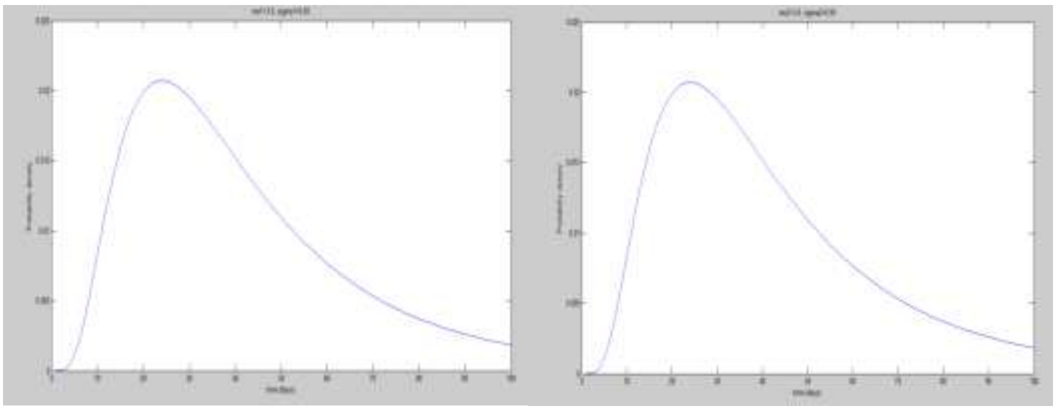
Fortuna rain – Coyolar flow:



(a) (b)

Fig. 42: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

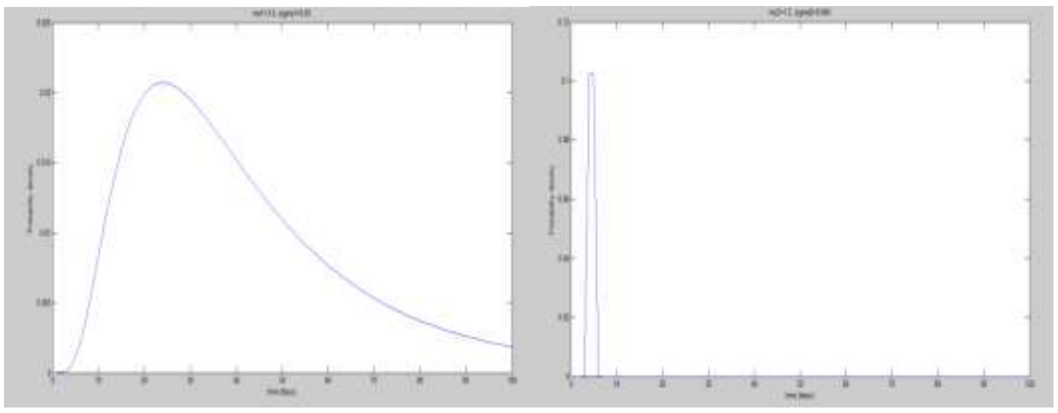
Guachipelín rain – Ponchote flow:



(a) (b)

Fig. 43: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

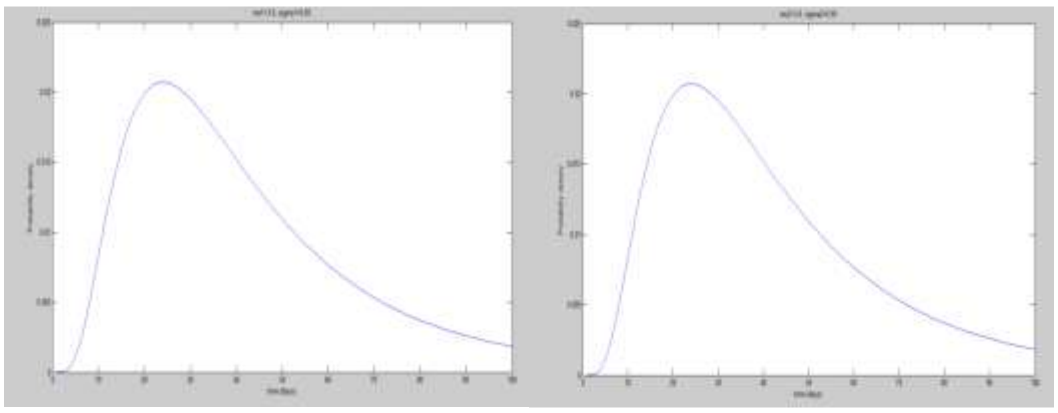
Guachipelín rain – Coyolar flow:



(a) (b)

Fig. 44: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

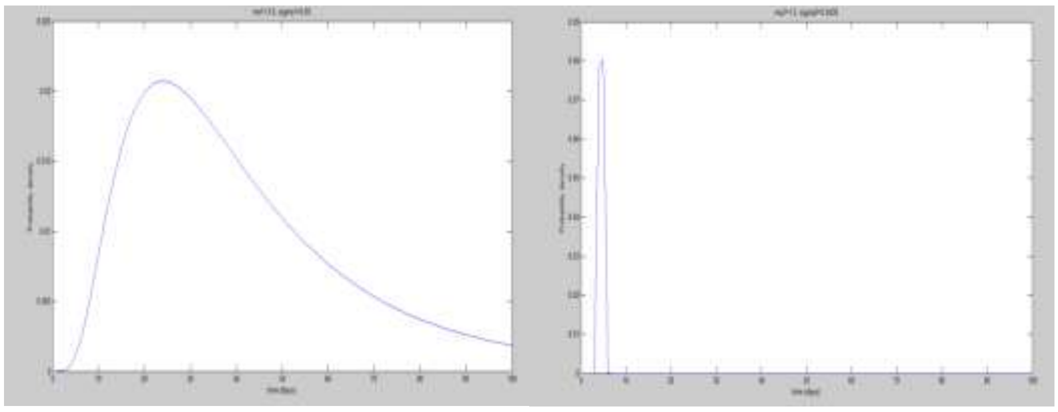
Hacienda La Flor rain – Ponchote flow:



(a) (b)

Fig. 45: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

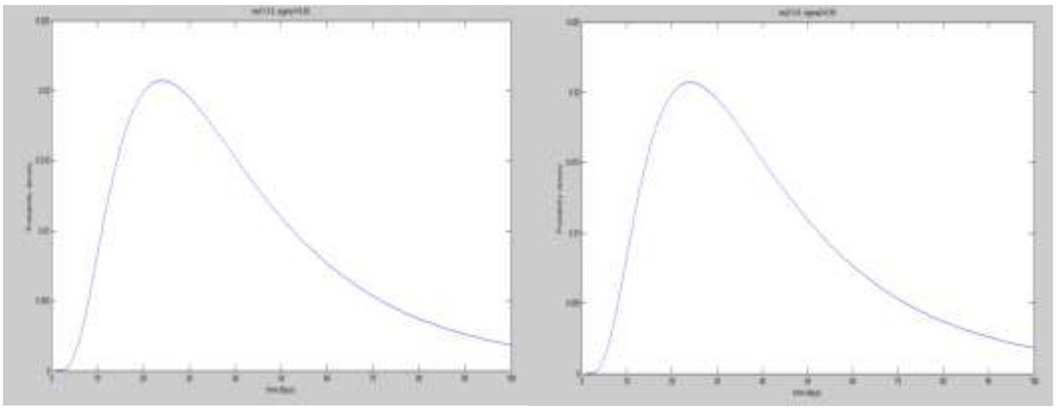
Hacienda La Flor rain – Coyolar flow:



(a) (b)

Fig. 46: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

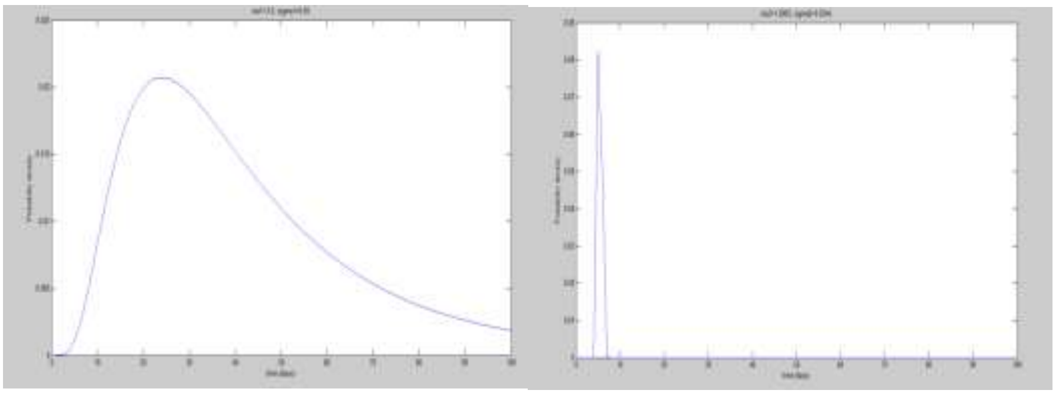
Hacienda Santa María rain – Ponchote flow:



(a) (b)

Fig. 47: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

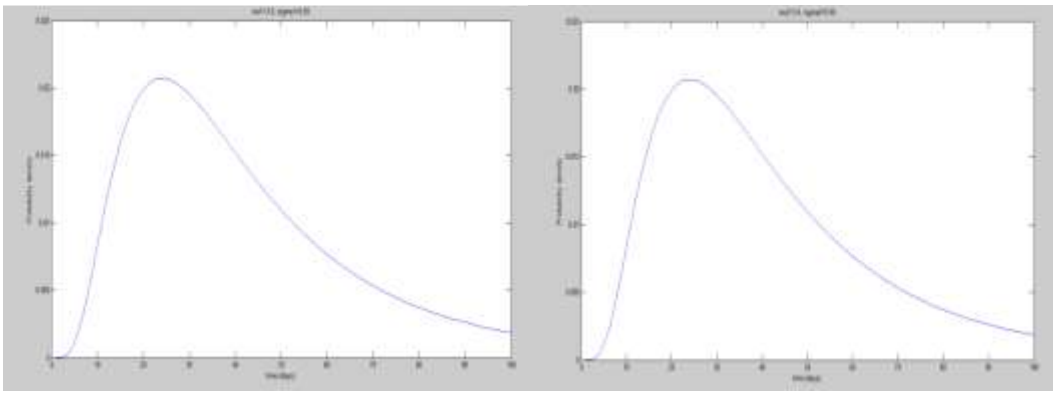
Hacienda Santa María rain – Coyolar flow:



(a) (b)

Fig. 48: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

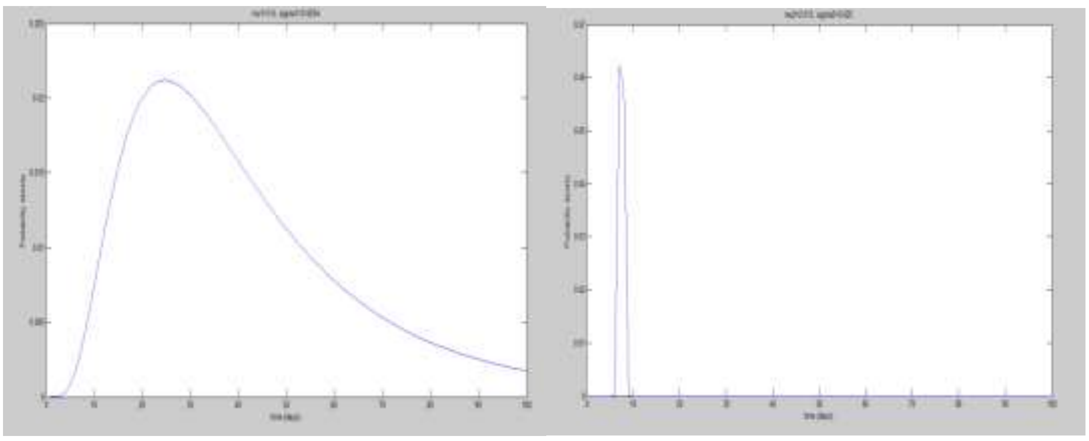
Liberia rain – Ponchote flow:



(a) (b)

Fig. 49: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

Liberia rain – Coyolar flow:



(a) (b)

Fig. 50: optimal Log-normal transfer function. Slow transfer function (a). Fast transfer function (b).

7. Conclusions

In this study, we implemented linear transfer function theory for modelling hydrological fluxes in the hydrological network of the Tempisque catchment in Costa Rica. We used a transfer function model to estimate the travel time water between rainfall and runoff in several points of the watershed. We were able to make a relatively good prediction of the discharge in terms of observed precipitation in the catchment using parametric transfer functions.

We used a transfer function composed of a fast and a slow reservoir. Six parametric forms of the transfer functions for the slow and fast reservoir were tested. The best results were obtained with the Log-normal, Gamma and Weibull transfer function. The results with the Log-normal were the most suitable, because the error between the modelled and real discharge was minimum.

As we have applied transfer functions (slow and fast) that represent travel time between nodes of the hydrological network, we can assess the mean and standard deviation of the travel time of water in parts of the Tempisque catchment.

The results showed that the slow transfer function has a variation of the travel time between 1.7451 and 3.6 days, with a mean of 3.3705. The standard deviation was between 0.6254 and 0.65 days, with a mean of 0.6484. In the case fast transfer function the travel time varied between 1.5 and 3.6 days, with a mean of 2.8749. The standard deviation of the fast transfer function was between 0.025 and 0.65 days, with a mean of 0.4583.

With this study we have realized the good ability to predict the simulated flow over the time through transfer functions models when the precipitation is known. Therefore this project could provide a good tool in order to fight against flood in the country.

References

- Abbott, M. B., Bathurst, J. C., Cunge, J. A., O'Connell, P.E. and Rasmussen, J. L. 1986 An introduction to the European Hydrology System (SHE). 2: Structure of a physically based distributed modelling system. Journal of Hydrology, 87, pp. 61-77.
- Alonso, A. 2013 Sustainable Management of Water Resources in Complex Engineered Water Systems: a Case Study in a NW Costa Rican Watershed. Department of Agricultural and Biological Engineering. University of Florida.
- Andréassian, V., Bergstrom, S., Chahinian, N., Duan, Q., Gusev, Y.M., Littlewood, I., Mathevet, T., Michel , C., Montanari, A., Moretti, G., Moussa, R., nasonova, O.N., O'Connor, K.M., Paquet, E., Perrin, C., Rousseau, A., Schaake, J., Wagener, T. and Xie, Z. 2006. Catalogue of the models used in MOPEX 2004/2005. IAHS Publication n°307, pp. 41-93.
- Beven K. J. 2001 Rainfall-runoff modelling: the primer. John Wiley & Sons, Ltd.
- Beven K. J. and Kirkby M. J. 1979 A physically based, variable contributing area model of basin hydrology. Hydrological Sciences Bulletin, pp. 43-69.
- Botter, G., Bertuzzo, E. and Rinaldo, A. 2011 Catchment residence and travel time distributions: The master equation. Geophysical Research Letters, 38(11), p. L11403.
- Brutsaert, W., 2005 Hydrology An Introduction. Cambridge University Press.
- Carrera, J., and Neuman S. P. 1986 Estimation of aquifer parameters under transient and steady state conditions, 1, Maximum likelihood method incorporating prior information, Water Resources Research. 22, pp. 199-210.
- Chang, M., Flannery, L.A. 1998. Evaluating the accuracy of rainfall catch by three different gauges. Journal of the American Water Resources Association 34: pp. 559-564.
- Coto, M. 2001 El Proyecto de Riego Arenal Tempisque. pp. 1-10.
- D'Odorico, P. and Rigon, R. 2003 Hillslope and channel contributions to the hydrologic response. WATER RESOURCES RESEARCH, 39(5), 1113, doi: 10.1029/2002WR001708.
- Gray, D.M. 1961 Synthetic hydrographs for small drainage areas. Proceedings of the ASCE, vol. 87, HY4. pp. 33-54.
- Hazell, P. Chakravorty, U., Dixon, J. and Celis, R. 2001. Monitoring Systems for Managing Natural Resources: Economics, Indicators and Environmental Externalities in a Costa Rican Watershed. pp. 1–156.
- Hunt, S. 2009 América latina en el siglo XX. ¿Se estrecharon las brechas o se ampliaron aún más? Desarrollo económico y bienestar. Homenaje a Máximo Vega-Centeno. Fondo Editorial de la Pontificia Universidad Católica del Perú.
- Huyer, W., Neumaier, A. 1999 Global Optimization by Multilevel Coordinate Search. J. Global Optim. 14. pp. 331-355.
- INEC 2013. Proyecciones de población al 30 de junio de 2013. < http://http://www.inec.go.cr/Web/Home/pagPrincipal.aspx>. [consultation: 05/03/2014].

- Jakeman, A. J., Littlewood, I. G. and Whitehead, P. G. 1990 Computation of the instantaneous unit hydrograph and identifiable component flows with application to two small upland catchments. Journal of Hydrology, 117, pp. 275-300.
- Jiménez, J.A., Calvo, J., Pizarro, F. and González, E. 2005 Conceptualization of Enviromental Flow in Costa Rica: Preliminary Determination for the Tempisque River. Unión Internacional para la Conservación de la Naturaleza, Oficina Regional para Mesoamérica (UICN/ORMA). pp. 1-40.
- Jiménez, J.A., Gonzalez, E. and Javier. M. V. 2001 Perspectives for the Integrated Management of The Tempisque River Basin, Costa Rica O. F. T. Studies, ed. 1–33.
- Jury, W.A. and Fluhler, H., 1992 Transport of Chemicals Through Soil--Mechanisms, Models, and Field Applications. Advances in Agronomy, 47, pp. 141–201.
- Kool, J. B., and Parker, J. C. 1988 Analysis of the inverse problem for transient unsaturated flow, Water Resources Research. 24, pp. 817-830.
- Lagarias, J. C., Reeds, J. A., Wright, M. H., Wright, P. E. 1998 Convergence Properties of the Nelder-Mead simplex algorithm in low dimensions, SIAM J. Optim., 9, pp. 112-147.
- Lambot, S., Javaux, M., Hupet, F., Vanclooster, M. 2002 A Global Multilevel Coordinate Search procedure for estimating the unsaturated soil hydraulic properties. Water Resour. Res. 38(11), 6/1-6/15.
- Lambot, S., Javaux, M., Hupet, F., Vanclooster, M. 2002 A global multilevel coordinate search procedure for estimating the unsaturated soil hydraulic properties, Water Resources Research. 1224, doi: 10.1029/2001WR001224.
- Melching, C. S., and Marquardt, J. S. 1997 Equations for Estimating Synthetic Unit-Hydrograph Parameter Values for Small Watersheds in Lake County, Illinois. U. S. Geological Survey. Urbana, Illinois.
- Moore, R.J. 2007 The PDM rainfall-runoff model. Hydrology and Earth System Sciences, pp. 483-499.
- Rabenstein, R. 1998 Transfer Function Models for Multidimensional Systems with Bounded Spatial Domains. Mathematical Modelling of Systems, Vol. 1, No. 1. 0-111.
- Ramírez, J. A. 2000 Prediction and Modeling of Flood Hydrology and Hydraulics. Chapter 11 of Inland Flood Hazards: Human, Riparian and Aquatic Communities Eds. Ellen Wohl; Cambridge University Press.
- Richter, D. 1995. Ergebnisse methodischer Untersuchungen zur Korrektur des systematischen Messfehlers des Hellmanniederschlagsmessers. Berichte des Deutschen Wetterdienstes 194: 93.
- Rinaldo, A., Marani, A. and Rignon, R. 1991 Geomorphological Dispersion. WATER RESOURCES RESEARCH, 27, pp. 513-525.
- Ritter, A., Hupet, F., Muñoz-Carpena, R., Vanclooster, M., Lambot, S. 2003 Using inverse methods for estimating soil hydraulic properties from field data as an alternative to direct methods. Agric. Water Manage,pp. 77-96.

- Ritter, A., Muñoz-Carpena, R., Regalado, C. M., Vanclooster, M., Lambot, S. 2004 Analysis of alternative measurement strategies for the inverse optimization of the hydraulic properties of a volcanic soil. Journal of Hydrology, 295, pp. 124-139.
- Sevruk, B. 2004. Niederschlag als Wasserkreislaufelement. ETH Zürich.
- Sugiura, K., Yang, D., Ohata, T. 2003 Systematic error aspects of gauge-measured solid precipitation in the Arctic, Barrow, Alaska. Geophysical Research Letters 30: 1-5.
- Taylor, A.B., Schwarz, H.E. 1952 Unit hydrograph lag and peak flow related to basin characteristics. Trans. Am. Geophys. Union 33, pp. 235-246.
- Wagener, T., Wheater, H. S. and Gupta, H. V. 2004 Rainfall-Runoff Modelling in Gauged and Ungauged Catchments, Imperial College press.
- Wagner A. 2009 Literature Study on the Correction of Precipitation Measurements. FutMon C1-Met-29(BY).
- Wheater H., Mcintyre, N. and Wagener, T. 2008 Calibration, uncertainty and regional analysis of conceptual rainfall-runoff models. In Wheater, H., Sorooshian, S. and Sharma, K. D. (Eds.) Hydrological Modelling in Arid and Semi-Arid Areas. Cambridge University Press.
- Wheater, H. S. 2008 Modelling Hydrological Processes in Arid and Semi Arid Areas: an Introduction. In Wheater, H., Sorooshian, S. and Sharma, K. D. (Eds.) Hydrological Modelling in Arid and Semi-Arid Areas. Cambridge University Press.
- Wheater, H. S., Jakeman, A. J. and Beven, K. J. 1993 Progress and directions in rainfall-runoff modelling. In Jakeman, A. J., Beck, M. B. and Mcaleer, M. J. (Eds.) Modelling Change in Environmental Systems.
- Whitehead P. G., Young P. C. and Hornberger G. M. 1979 A systems model of streamflow and water quality in the Bedford – Ouse River. 1. Streamflow modelling. Water Resources Research, 13, pp. 1155 – 1169.
- Woo, M., Steer, P. 1979. Measurement of trace rainfall at a high arctic site. Arctic 32: pp. 80-84.
- World Meteorological Organization (WMO) 1994. Guide to hydrological practices. WMO report 168.
- World Meteorological Organization (WMO) 2008. Guide to meteorological instruments and methods of observation. WMO-8 8 1-681.
- Yang, D., Ohata, T. 2001 A bias-corrected Siberian regional precipitation climatology. Journal of Hydrometeorology: 2, pp. 122-139.
- Young P. C. 1975 Recursive approaches to time series analysis. Buelletin of the Institute of Math. Appl. 10, pp. 209 – 224.
- Young P. C. 2001 Data-based mechanistic modelling and validation of rainfall-flow processes. Centre for Research on Environmental Systems and Statistics, Institute of Environmental and Natural Sciences, Lancaster University, Lancaster.

ANNEXE: Implementation in Matlab

The routines carried out in Matlab in order to calibrate the transfer function model are very extensive. In this section we will explain some of the routines and also the functions used.

The codes of the transfer functions used in this study are shown as follows:

Simple Transfer Function

tf = A – B.*exp(-t/T)

Normal Probabilty Density Function

tf = normpdf(t,Mu,sigma)

Lognormal Probabilty Density Function

tf = lognpdf(t,Mu,sigma)

Gamma Probabilty Density Function

tf = gampdf(t,A,B)

Beta Probabilty Density Function

tf = betapdf(t,A,B)

Weibull Probabilty Density Function

tf = wblpdf(t,A,B)

The simulated discharge was calculated through the mathematical convolution between the precipitation and the transfer function.

y = convolution(a,b)

The convolution function calculate the product convolution between the vectors a and b in the context of applying a filter to a series. The vector b acts as filter, and the length of the vector result is the same that vector a.

To establish the objective function (OF), difference between the measured and modelled discharges, we have applied the next routine:

```
function [OF] = obj_fun(X0)
```

The vector determination of hydrologic parameters:

```
tfm_parameters = PARAM;

i2 = 1;

for i1 = 1:length(tfm_parameters)

   if isnan(tfm_parameters(i1))

      tfm_parameters(i1) = X0(i2)

      i2 = i2+1;

   end

end
```

To establish the initial variables:

```
Ymodel = []; ERR = []; Wvect = [];
```

The calculation of the direct problem for the given the parameters vector:

```
[Ymodel] = tfm(t,rain,tfm_parameters);
```

The objective function is calculated only if the direct problem has converged.

```
if ~isempty(Ymodel)

   WEIGHT = [1];
```

“Weight” is only a simplification to save memory, only valid if all elements of Wvect are equal.

The Calculation of the residuals vector:

```
    ERR = Ymeas-Ymodel;
```

The Calculation of the objective function:

```
    OF = ERR'*WEIGHT*ERR;

else
   OF = 1;
end
```

The calibration implementation was carried out through the next routines:

The parameters data were established through the next function:

```
Function[PARAM,U,V,DIS1,Qp,NM,smax,nf,stop,iinit,local,gamma]
= param_data()
```

where:

-PARAM: referred to the parameters of every transfer function, in our case Mu and Sigma (Log-normal).

-U and V: define the parameters space in which the optimization will be carried out. Being U the lower bound and V the upper.

-DIS1: define the discretization of parameters space, concretely the number of values between lower and upper bounds.

-Qp: define values of fitted parameters to use as constants.

-NM: define the different parameters couple for which the objective function will be calculated and plotted.

-smax: is the number of levels, smax governs the relative amount of global versus local search.

-nf: is the maximum number of function evaluations.

-stop: is the integer defining stopping test.

-iinit: is the parameter defining the initialization list.

-local: is the maximal number of steps in local search.

-gamma: is the stopping criterion for local search.

The Global Multilevel Coordinate Search Function was defines as:

```
[xbest,fbest,xmin,fmi,ncall,ncloc,flag] =

mcs('feval','obj_fun',U,V,0,smax,nf,stop,iinit,local,gamma,one
s(np,np))
```

During the calibration the Jacobian, the covariance matrix (C), and the correlation matrix (A) are needed to carry out the calibration. The routines developed in Matlab in order to make these calculations were the follow:

To calculate the Jacobian (J)

```
J = zeros(n,p);
for k = 1:p
   delta = zeros(p,1);
   delta(k) = xbest2(k)*0.01;
   [OF] = obj_fun(xbest2+delta);
   if OF == 1
      [OF] = obj_fun(xbest2-delta);
   end
   J(:,k) = (Ymodel-Ymodelfit)/delta(k);
End
```

To calculate the parameters covariance matrix (C)

```
S = chol(WEIGHT);   %Cholesky decomposition
Jw = S*J;
ERRw = S*ERR;
ERR_var = (ERRw'*ERRw)/(n-p);
C = ERR_var*inv(Jw'*Jw);
Cii = diag(C);
```

To calculate the parameters correlation matrix (A)

```
Ci = Cii*ones(1,p);
Cj = Ci';
A = C./(sqrt(Ci).*sqrt(Cj));
```

Local optimization:
As we have explained before, the global optimization is combined with the local optimization through the Nelder Mead. We have used the next function.

```
x = fminsearch(fun,x0)
```

MULTILEVEL COORDINATE SEARCH FUNCTION:

The Global Multilevel Coordinate Search was programmed as follow:

Input:

-fcn: 'fun'   name of function fun(data,x), x an n-vector
-data: data vector (or other data structure)
-[u,v]:   box in which the optimization is carried out (u, v n-vectors)
- prt: print level
    prt = 0: no printing
    prt = 1: # sweep, minimal nonempty level, # f-calls, best point and function value (default)
    prt > 1: only meaningful for test functions with known global minimizes in addition levels and function values of boxes containing the global minimizes of a test function.
- smax: number of levels (default: 5*n+10)
-nf: maximum number of function evaluations (default: 50*n^2)
- stop: stop(1) in]0,1[: relative error with which the known global minimum of a test function should be found.
        stop(2) = fglob known global minimum of a test function
        stop(3) = safeguard parameter for absolutely small
    fglob:
        stop(1) >= 1: the program stops if the best function value has not been improved for stop(1) sweeps.
        stop(1) = 0: the user can specify a function value that should be reached
        stop(2) = function value that is to be achieved
      (default: stop = 3*n)
- iinit: parameter defining the initialization list
    = 0   corners and midpoint (default for finite u,v)
    = 1   safeguarded version *default otherwise)
    = 2   5u/6 + v/6, u/6 + 5v/6 and midpoint
    = 3   initialization list with line searches otherwise self-defined init. list (to be stored in init0.m)

Output:

xbest(1:n):   current best point
fbest: function value at xbest
xmin: matrix with n rows; the columns are the points in the 'shopping basket' (i.e. good points resp. local minimizers)
fmi: function values corresponding to the 'shopping basket';
        fmi(i) is the function value at xmin(:,i)
ncall: number of function evaluations
ncloc: number of function evaluations used for local search
flag: specifies which stopping criterion has been used
      = 0 a (known) global minimum fglob of a test function has been found with the required relative error
    Relerr:
      = 1 the division procedure has been completed
      = 2 the maximum number nf of function calls has been reached without finding a known minimum with the required relative error or completing the division procedure
      = 3 stop(1) sweeps without progress (for stop(1) >= 1)

function
[xbest,fbest,xmin,fmi,ncall,ncloc,flag]=mcs(fcn,data,u,v,prt,smax,nf,stop,iinit,local,gamma,hess)

Global variables
global foptbox nbasket nboxes ncall nglob nsweep nsweepbest optlevel record xglob xloc

foptbox(1:nglob): function value(s) of the box(es) containing the (a)global minimizer of a test function
 -nbasket: counter for boxes in the 'shopping basket'
 -nboxes: counter for boxes not in the 'shopping basket'
 -nglob: number of global minimizers of a test function
- nloc: (for local ~= 0) counter of points that have been used as starting points for a local search
- nsweep: sweep counter
 -nsweepbest: number of sweep in which fbest was updated for the last time
 -optlevel: level(s) of the box(es) containing the (a) global minimum of a test function
 -record(1:smax-1) record(i) points to the best non-split box at level i(record list)
- xglob(1:n,1:nglob) xglob(:,i), i=1:nglob, are the global minimizers of a test function in [u,v]
-xloc(1:n,:)   (for local ~= 0) columns are the points that have been used as starting points for local search

```
n = length(u);
```

To check box bounds:
```
if ~isempty(find(v<u))
  error('incompatible box bounds')
elseif ~isempty(find(u==v))
  error('degenerate box bound')
end
```

Default values for the input parameters:
```
if nargin < 5, prt = 1; end
if nargin < 6, smax = 5*n+10; end
if nargin < 7, nf = 50*n^2; end
if nargin < 8, stop = 3*n; end
if nargin < 9,
  if isempty(find(isinf(u))) & isempty(find(isinf(v)))
    iinit = 0;
  else
    iinit = 1;
  end
end
if nargin < 10, local = 50; end
if nargin < 11, gamma = eps; end
if nargin < 12, hess = ones(n,n); end


xmin=[];fmi=[];      % avoid warnings in Matlab
```

The initial values for the numbers of function calls (total number/local search):
```
ncall = 0;
ncloc = 0;
```

Some parameters needed for initializing large arrays:
```
step1 = 10000;
```

```
step = 1000;
dim = step1;
```

The initialization of some large arrays:

```
isplit = zeros(1,step1);
level = zeros(1,step1);
ipar = zeros(1,step1);
ichild = zeros(1,step1);
f = zeros(2,step1);
z = zeros(2,step1);
nogain = zeros(1,step1);
```

Definition of the initialization list:

```
if iinit == 0
  x0(:,1) = u;
  x0(:,2) = (u+v)/2;
  x0(:,3) = v;
  l = 2*ones(n,1);
  L = 3*ones(n,1);
elseif iinit == 1
  for i = 1:n
    if u(i) >= 0
      x0(i,1) = u(i); [x0(i,2),x0(i,3)] = subint(u(i),v(i));x0(i,2) =
0.5*(x0(i,1)+x0(i,3));
    elseif v(i) <= 0
      x0(i,3) = v(i); [x0(i,2),x0(i,1)] = subint(v(i),u(i));x0(i,2) =
0.5*(x0(i,1)+x0(i,3));
    else
      x0(i,2) = 0; [xi,x0(i,1)] = subint(0,u(i)); [xi,x0(i,3)] = subint(0,v(i));
    end
  end
  l = 2*ones(n,1);
  L = 3*ones(n,1);
elseif iinit == 2
  x0(:,1) = (5*u + v)/6;
  x0(:,2) = 0.5*(u + v);
  x0(:,3) = (u + 5*v)/6;
  l = 2*ones(n,1);
  L = 3*ones(n,1);
elseif iinit == 3
  [x0,f0,l,L,istar,ncall1] = initlist(fcn,data,u,v);
  ncall = ncall + ncall1;
else
  init0     %self-defined initialization list
  for i=1:size(x0,2)
    if ~isempty(find(x0(:,i)<u)) | ~isempty(find(x0(:,i)>v))
      error('incorrect initialization list')
    end
  end
end
```

To check whether there are infinities in the initialization list:

```
if ~isempty(find(isinf(x0))), error('infinities in ititialization list'), end

if iinit ~= 3
  [f0,istar,ncall1] = init(fcn,data,x0,l,L,n);
  ncall = ncall + ncall1;
end
```

Definition of the base vertex of the original box:
```
for i = 1:n
  x(i) = x0(i,l(i));
end
```

Definition of the opposite vertex v1 of the original box:
```
for i = 1:n
  if abs(x(i)-u(i)) > abs(x(i)-v(i))
    v1(i) = u(i);
  else
    v1(i) = v(i);
  end
end
```

Initialization of the record list, the counters nboxes, nbasket, m and nloc, xloc and the output flag:
```
record = zeros(smax-1,1);
nboxes = 1;
nbasket = 0;
nbasket0 = 0;
nsweep = 0;
m = n;
record(1) = 1;
nloc = 0;
xloc = [];
flag = 1;

[ipar,level,ichild,f,isplit,p,xbest,fbest] = initbox(x0,f0,l,L,istar,u,v,prt);
```

Generates the boxes in the initialization procedure:
```
f0min = fbest;
if stop(1) > 0 & stop(1) < 1
  flag = chrelerr(fbest,stop);
elseif stop(1) == 0
  flag = chvtr(fbest,stop(2));
end
if ~flag,return,end
```

if the (known) minimum function value fglob has been found with the required tolerance, flag is set to 0 and the program is terminated

```
s = strtsw(smax,level,f(1,:));
```

The vector record is updated, and the minimal level s containing non-split boxes is computed
```
nsweep = nsweep + 1


while s < smax & ncall + 1 <= nf
  par = record(s);
  [n0,x,y,x1,x2,f1,f2] = vertex(par,n,u,v,v1,x0,f0,ipar,isplit,ichild,z,f,l,L);
  if s > 2*n*(min(n0)+1)

    [isplit(par),z(2,par)] = splrnk(n,n0,p,x,y);
    splt = 1;
  else
    if nogain(par)
      splt = 0;
    else
```

```
      [e,isplit(par),z(2,par)] = exgain(n,n0,l,L,x,y,x1,x2,f(1,par),f0,f1,f2);
      fexp = f(1,par) + min(e);
      if fexp < fbest
        splt = 1;
      else
        splt = 0;
        nogain(par) = 1;
      end
    end
  end
  if splt == 1
    i = isplit(par);
    level(par) = 0;
    if z(2,par) == Inf
      m = m + 1;
      z(2,par) = m;
      [xbest,fbest,f0(:,m),xmin,fmi,ipar,level,ichild,f,flag,ncall1] =
splinit(fcn,data,i,s,smax,par,x0,n0,u,v,x,y,x1,x2,L,l,xmin,fmi,ipar,level,ichild
,f,xbest,fbest,stop,prt);
      ncall = ncall + ncall1;
    else
      z(1,par) = x(i);
      [xbest,fbest,xmin,fmi,ipar,level,ichild,f,flag,ncall1] =
split(fcn,data,i,s,smax,par,n0,u,v,x,y,x1,x2,z(:,par),xmin,fmi,ipar,level,ichild
,f,xbest,fbest,stop,prt);
      ncall = ncall + ncall1;
    end
    if nboxes > dim
```

If the pre-assigned size of the `large' arrays has already been exceeded, these arrays are made larger:

```
      isplit(nboxes+1:nboxes+step) = zeros(1,step);
      level(nboxes+1:nboxes+step) = zeros(1,step);
      ipar(nboxes+1:nboxes+step) = zeros(1,step);
      ichild(nboxes+1:nboxes+step) = zeros(1,step);
      z(:,nboxes+1:nboxes+step) = zeros(2,step);
      nogain(nboxes+1:nboxes+step) = zeros(1,step);
      f(:,nboxes+1:nboxes+step) = zeros(2,step);
      dim = nboxes + step;
    end
    if ~flag,break,end
  else
    if s + 1 < smax
      level(par) = s + 1;
      updtrec(par,s+1,f(1,:));
    else
      level(par) = 0;
      nbasket = nbasket + 1;
      xmin(:,nbasket) = x;
      fmi(nbasket) = f(1,par);
    end
    if prt > 1
      [w1,w2] = bounds(n,n0,x,y,u,v);
      iopt = [];
      for iglob = 1:nglob
        if w1 <= xglob(:,iglob) & xglob(:,iglob) <= w2
          iopt = [iopt, iglob];
        end
        for iglob = 1:length(iopt)
          optlevel(iopt(iglob)) = s + 1;
        end
      end
    end
  end   s = s + 1;
  while s < smax
    if record(s) == 0
      s = s + 1;
    else
      break
    end
  end
  if s == smax
    if local,
      [fmi(nbasket0+1:nbasket),j] = sort(fmi(nbasket0+1:nbasket));
      xmin(:,nbasket0+1:nbasket) = xmin(:,nbasket0+j);
      xmin0 = [];
      fmi0 = [];
      for j = nbasket0+1:nbasket
        x = xmin(:,j);
        f1 = fmi(j);
        chkloc;
        if loc,
          addloc;
          [xbest,fbest,xmin,fmi,x,f1,loc,flag,ncall1] =
basket(fcn,data,x,f1,xmin,fmi,xbest,fbest,stop,nbasket0);
          ncall = ncall + ncall1;
          if ~flag,break,end
          if loc,
            [xmin1,fmi1,nc,flag] = lsearch(fcn,data,x,f1,f0min,u,v,nf-
ncall,stop,local,gamma,hess);
            ncall = ncall + nc;
            ncloc = ncloc + nc;
            if fmi1 < fbest
              xbest = xmin1;
              fbest = fmi1;
              nsweepbest = nsweep;
              if ~flag
                nbasket0 = nbasket0 + 1;
                nbasket = nbasket0;
                xmin(:,nbasket) = xmin1;
                fmi(nbasket) = fmi1;
                break
              end
              if stop(1) > 0 & stop(1) < 1
                flag = chrelerr(fbest,stop);
              elseif stop(1) == 0
                flag = chvtr(fbest,stop(2));
              end
              if ~flag,return,end
            end
            [xbest,fbest,xmin,fmi,loc,flag,ncall1] =
basket1(fcn,data,xmin1,fmi1,xmin,fmi,xbest,fbest,stop,nbasket0);
            ncall = ncall + ncall1;
            if ~flag,break,end
            if loc,
              nbasket0 = nbasket0 + 1;
              xmin(:,nbasket0) = xmin1;
              fmi(nbasket0) = fmi1;
              fbestloc;
              if ~flag,
                nbasket = nbasket0; break
              end
            end
          end
        end
      end
      nbasket = nbasket0;
      if ~flag,break,end
    end
    s = strtsw(smax,level,f(1,:));
    if prt,
      if nsweep == 1
        fprintf('nsw  minl  ');
        if prt > 1
          fprintf('optl    fopt       ')
        end
        fprintf('nf     fbest        xbest\n')
      end
      minlevel=s;
      fprintf('%3i  %3i',nsweep,minlevel);
      if prt > 1
        fprintf('  %3i',optlevel);fprintf('  %10.3e',foptbox);
      end
      fprintf('  %5i  %10.3e',ncall,fbest);
      fprintf('  %10.4f',xbest);
      fprintf(1,'\n');
    end
    if stop(1) > 1
      if nsweep - nsweepbest >= stop(1),flag = 3; return,end
    end
    nsweep = nsweep + 1;
  end
end
if ncall >= nf
  flag = 2;
end
if local,
  if length(fmi) > nbasket
    xmin(:,nbasket+1:length(fmi)) = [];
    fmi(nbasket+1:length(fmi)) = [];
  end
end
```

BASKET FUNCTION:

The Basket function checks whether a candidate for local search lies in the 'domain of attraction' of a point in the 'shopping basket'.

```
function [xbest,fbest,xmin,fmi,nbasket,loc,flag] =
basket(fcn,data,x,f,xmin,fmi,xbest,fbest,stop,nbasket)

global nsweep nsweepbest
loc = 1;
flag = 1;
ncall = 0;
if ~nbasket, return, end
for k = 1:nbasket
  dist(k) = norm(x - xmin(:,k));
end
[dist1,ind] = sort(dist);
for k = 1:nbasket
  i = ind(k);
  if fmi(i) <= f
    p = xmin(:,i) - x;
    y1 = x + 1/3*p;
    f1 = feval(fcn,data,y1);
    ncall = ncall + 1;
    if f1 <= f
      y2 = x + 2/3*p;
      f2 = feval(fcn,data,y2);
      ncall = ncall + 1;
      if f2 > max(f1,fmi(i))
        if f1 < f
          x = y1;
          f = f1;
          if f < fbest
            fbest = f;
            xbest = x;
            nsweepbest = nsweep;
            if stop(1) > 0 & stop(1) < 1
              flag = chrelerr(fbest,stop);
            elseif stop(1) == 0
              flag = chvtr(fbest,stop(2));
            end
            if ~flag,return,end
          end
        end
      else
        if f1 < min(f2,fmi(i))
          f = f1;
          x = y1;
          if f < fbest
            fbest = f;
            xbest = x;
            nsweepbest = nsweep;
            if stop(1) > 0 & stop(1) < 1
              flag = chrelerr(fbest,stop);
            elseif stop(1) == 0
              flag = chvtr(fbest,stop(2));
            end
            if ~flag,return,end
          end
        elseif f2 < min(f1,fmi(i))
          f = f2;
          x = y2;
          if f < fbest
            fbest = f;
            xbest = x;
            nsweepbest = nsweep;
            if stop(1) > 0 & stop(1) < 1
              flag = chrelerr(fbest,stop);
            elseif stop(1) == 0
              flag = chvtr(fbest,stop(2));
            end
            if ~flag,return,end
          end
        else
          loc = 0;break
        end
      end
    end
  end
end
```

INIT FUNCTION:

The Init function computes the function values corresponding to the initialization list and the pointer istar to the final best point x^* of the init list.

```
function [f0,istar,ncall] = init(fcn,data,x0,l,L,n)
function [f0,istar,ncall] = init(fcn,data,x0,l,L,n)
ncall = 0;
for i = 1:n
  x(i) = x0(i,l(i));
end
x = x';
f1 = feval(fcn,data,x);
f0(l(1),1) = f1;
ncall = ncall + 1;
for i = 1:n
  istar(i) = l(i);
  for j = 1:L(i)
    if j == l(i)
      if i ~= 1
        f0(j,i) = f0(istar(i-1),i-1);
      end
    else
      x(i) = x0(i,j);
      f0(j,i) = feval(fcn,data,x);
      ncall = ncall + 1;
      if f0(j,i) < f1
        f1 = f0(j,i);
        istar(i) = j;
      end
    end
  end
  x(i) = x0(i,istar(i));
end
```

INITLIST FUNCTION:

The Initlist function generates an initialization list with the aid of line searches.

```
function [x0,f0,l,L,istar,ncall] = initlist(fcn,data,u,v)
ncall = 0;
nloc = 5;
small = 0.1;
smaxls = 25;
n = length(u);
x = min(max(u,0),v);
f = feval(fcn,data,x);
ncall = ncall + 1;
for i = 1:n
  alist = 0;
  flist = f;
  p = zeros(n,1);
  p(i) = 1;
  [alist,flist,nfls] = gls(fcn,data,u,v,x,p,alist,flist,nloc,small,smaxls);
  ncall = ncall + nfls;
  [alist1,flist1] = lspost(alist,flist);
  if isempty(find(alist1==0))
    alist1 = [alist1 0];
    flist1 = [flist1 f];
  end
  if length(alist1) < 3
    if isempty(find(alist1==alist(length(alist))))
      alist1 = [alist1 alist(length(alist))];
      flist1 = [flist1 flist(length(alist))];
    end
    if length(alist1) < 3
      if isempty(find(alist1==alist(1)))
        alist1 = [alist1 alist(1)];
        flist1 = [flist1 flist(length(alist))];
      end
      if length(alist1) < 3
        k = round((1+length(alist))/2);
        alist1 = [alist1 alist(k)];
        flist1 = [flist1 flist(k)];
      end
    end
  end
  [alist,ind] = sort(alist1);
  flist = flist1(ind);
  l(i) = find(alist == 0);
  [f1,istar(i)] = min(flist);
  L(i) = length(alist);
  x0(i,1:L(i)) = alist + x(i);
  f0(1:L(i),i) = flist';
  x(i) = x0(i,istar(i));
  f = feval(fcn,data,x);
end
```

CHRELERR FUNCTION:

The Chrelerr function checks whether the required tolerance for a test function with known global minimum has already been achieved.

```
function flag = chrelerr(fbest,stop)

fglob = stop(2);
if fbest - fglob <= max(stop(1)*abs(fglob),stop(3))
  flag = 0;
else
  flag = 1;
end
```

STRTSW FUNCTION:

The STRTSW function updates the record list for starting a new sweep and computes the lowest level containing non-split boxes.

```
function s = strtsw(smax,level,f)

record = zeros(smax-1,1);
s = smax;
for j = 1:nboxes
  if level(j) > 0
    if level(j) < s
      s = level(j);
    end
    if ~record(level(j))
      record(level(j)) = j;
    elseif f(j) < f(record(level(j)))
      record(level(j)) = j;
    end
  end
end
```

SPLRNK FUNCTION:

The SPLRNK function determines the splitting index and splitting value for splitting a box by rank.

```
function [isplit,splval] = splrnk(n,n0,p,x,y)

isplit = 1;
n1 = n0(1);
p1 = p(1);
for i = 2:n
 if n0(i) < n1 | (n0(i) == n1 & p(i) < p1)
  isplit = i;
  n1 = n0(i);
  p1 = p(i);
 end
end
if n1 > 0
 splval = split2(x(isplit),y(isplit));
else
 splval =  Inf;
end
```

EXGAIN FUNCTION:

The exgain function determines the splitting index, the splitting value and the expected gain vector e for (potentially) splitting a box by expected gain.

```
function [e,isplit,splval] = exgain(n,n0,l,L,x,y,x1,x2,fx,f0,f1,f2)

emin = Inf;  % initialization
for i = 1:n
  if n0(i) == 0
    e(i) = min(f0(1:L(i),i)) - f0(l(i),i);
    if e(i) < emin
      emin = e(i);
      isplit = i;
      splval = Inf;
    end
  else
    z1 = [x(i) x1(i) x2(i)];
    z2 = [0 f1(i) - fx f2(i) - fx];
    d = polint(z1,z2);
    [eta1,eta2] = subint(x(i),y(i));
    % safeguard against splitting too close to x(i)
    xi1 = min(eta1,eta2);
    xi2 = max(eta1,eta2);
    z = quadmin(xi1,xi2,d,z1);
    e(i) = quadpol(z,d,z1);
    if e(i) < emin
      emin = e(i);
      isplit = i;
      splval = z;
    end
  end
end
```

VETEX FUNCTION:

The Vetex function computes the base vertex x and the opposite vertex y of the box # j of MCS and the 'neighboring vertices' x1 and x2 and their functionvalues f1 and f2 needed for separable quadratic interpolation.

```
function [n0,x,y,x1,x2,f1,f2] =
vertex(j,n,u,v,v1,x0,f0,ipar,isplit,ichild,z,f,l,L)

x = Inf*ones(n,1);
y = Inf*ones(n,1);
x1 = Inf*ones(n,1);
x2 = Inf*ones(n,1);
f1 = zeros(n,1);
f2 = zeros(n,1);
n0 = zeros(n,1);
fold = f(1,j);
m = j;
while m > 1
  i = abs(isplit(ipar(m)));
  n0(i) = n0(i) + 1;
  if ichild(m) == 1
    if x(i) == Inf | x(i) == z(1,ipar(m))
      [x(i),x1(i),x2(i),f1(i),f2(i)] =
vert1(2,z(:,ipar(m)),f(:,ipar(m)),x1(i),x2(i),f1(i),f2(i));
    else
      [f1,f2,fold] = updtf(n,i,x1,x2,f1,f2,fold,f(1,ipar(m)));
      [x1(i),x2(i),f1(i),f2(i)] =
vert2(1,x(i),z(:,ipar(m)),f(:,ipar(m)),x1(i),x2(i),f1(i),f2(i));
    end
  elseif ichild(m) >= 2
    [f1,f2,fold] = updtf(n,i,x1,x2,f1,f2,fold,f(1,ipar(m)));
    if x(i) == Inf | x(i) == z(2,ipar(m))
      [x(i),x1(i),x2(i),f1(i),f2(i)] =
vert1(1,z(:,ipar(m)),f(:,ipar(m)),x1(i),x2(i),f1(i),f2(i));
    else
      [x1(i),x2(i),f1(i),f2(i)] =
vert2(2,x(i),z(:,ipar(m)),f(:,ipar(m)),x1(i),x2(i),f1(i),f2(i));
    end
  end
  if 1 <= ichild(m) & ichild(m) <= 2 & y(i) == Inf
    y(i) = split1(z(1,ipar(m)),z(2,ipar(m)),f(1,ipar(m)),f(2,ipar(m)));
  end
  if ichild(m) < 0
    if u(i) < x0(i,1)
      j1 = ceil(abs(ichild(m))/2);
      j2 = floor(abs(ichild(m))/2);
      if (abs(ichild(m))/2 < j1  & j1 > 1) | j1 == L(i)
        j3 = -1;
      else
        j3 = 1;
      end
    else
      j1 = floor(abs(ichild(m))/2) + 1;
      j2 = ceil(abs(ichild(m))/2);
      if abs(ichild(m))/2 + 1 > j1 & j1 < L(i)
        j3 = 1;
      else
        j3 = -1;
      end
    end
    if isplit(ipar(m)) < 0
      k = i;
    else
      k = z(2,ipar(m));
    end
    if j1 ~= l(i) | (x(i) ~= Inf & x(i) ~= x0(i,l(i)))
      [f1,f2,fold] = updtf(n,i,x1,x2,f1,f2,fold,f0(l(i),k));
    end
    if x(i) == Inf | x(i) == x0(i,j1)
      x(i) = x0(i,j1);
      if x1(i) == Inf
        [x1(i),x2(i),f1(i),f2(i)] =
vert3(j1,x0(i,:),f0(:,k),L(i),x1(i),x2(i),f1(i),f2(i));
      elseif x2(i) == Inf & x1(i) ~= x0(i,j1+j3)
        x2(i) = x0(i,j1+j3);
        f2(i) = f2(i) + f0(j1+j3,k);
      elseif x2(i) == Inf
        if j1 ~= 1 & j1 ~= L(i)
          x2(i) = x0(i,j1-j3);
          f2(i) = f2(i) + f0(j1-j3,k);
        else
          x2(i) = x0(i,j1+2*j3);
          f2(i) = f2(i) + f0(j1+2*j3,k);
        end
      end
    else
      if x1(i) == Inf
        x1(i) = x0(i,j1);
        f1(i) = f1(i) + f0(j1,k);
        if x(i) ~= x0(i,j1+j3)
          x2(i) = x0(i,j1+j3);
          f2(i) = f2(i) + f0(j1+j3,k);
        end
      elseif x2(i) == Inf
        if x1(i) ~= x0(i,j1)
          x2(i) = x0(i,j1);
          f2(i) = f2(i) + f0(j1,k);
        elseif x(i) ~= x0(i,j1+j3)
          x2(i) = x0(i,j1+j3);
          f2(i) = f2(i) + f0(j1+j3,k);
        else
          if j1 ~= 1 & j1 ~= L(i)
            x2(i) = x0(i,j1-j3);
            f2(i) = f2(i) + f0(j1-j3,k);
          else
            x2(i) = x0(i,j1+2*j3);
            f2(i) = f2(i) + f0(j1+2*j3,k);
          end
        end
      end
    end
    if y(i) == Inf
      if j2 == 0
        y(i) = u(i);
      elseif j2 == L(i)
        y(i) = v(i);
      else
        y(i) = split1(x0(i,j2),x0(i,j2+1),f0(j2,k),f0(j2+1,k));
      end
    end
  end
  m = ipar(m);
end
for i = 1:n
  if x(i) == Inf
    x(i) = x0(i,l(i));
    [x1(i),x2(i),f1(i),f2(i)] =
vert3(l(i),x0(i,:),f0(:,i),L(i),x1(i),x2(i),f1(i),f2(i));
  end
  if y(i) == Inf
     y(i) = v1(i);
  end
end
```

SPLINIT FUNCTION:
The Splinit function splits box # par at level s according to the initialization list in the ith coordinate and inserts its children and their parameters in the list.

```
function [xbest,fbest,f0,xmin,fmi,ipar,level,ichild,f,flag,ncall] =
splinit(fcn,data,i,s,smax,par,x0,n0,u,v,x,y,x1,x2,L,l,xmin,fmi,xbest,fbest,ipar,
level,ichild,f,stop,prt)

splinit(fcn,data,i,s,smax,par,x0,n0,u,v,x,y,x1,x2,L,l,xmin,fmi,ipar,level,ichild
,f,xbest,fbest,stop,prt)

global nbasket nboxes nglob nsweep nsweepbest record xglob xloc
ncall = 0;
n = length(x);
f0 = zeros(max(L),1);
flag = 1;
if prt > 1
  [w1,w2] = bounds(n,n0,x,y,u,v);
  iopt = [];
  for iglob = 1:nglob
    if w1 <= xglob(:,iglob) & xglob(:,iglob) <= w2
      iopt = [iopt, iglob];
    end
  end
end
for j=1:L(i)
  if j ~= l(i)
    x(i) = x0(i,j);
    f0(j) = feval(fcn,data,x);
    ncall = ncall + 1;
    if f0(j) < fbest
      fbest = f0(j);
      xbest = x;
      nsweepbest = nsweep;
      if stop(1) > 0 & stop(1) < 1
        flag = chrelerr(fbest,stop);
      elseif stop(1) == 0
        flag = chvtr(fbest,stop(2));
      end
      if ~flag,return,end
    end
  else
    f0(j) = f(1,par);
  end
end
[fm,i1] = min(f0);
if i1 > 1
```

```
  splval1 = split1(x0(i,i1-1),x0(i,i1),f0(i1-1),f0(i1));
else
  splval1 = u(i);
end
if i1 < L(i)
  splval2 = split1(x0(i,i1),x0(i,i1+1),f0(i1),f0(i1+1));
else
  splval2 = v(i);
end
if s + 1 < smax
  nchild = 0;
  if u(i) < x0(i,1)
    nchild = nchild + 1;
    nboxes = nboxes + 1;
    [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] = genbox(par,s+1,-
nchild,f0(1));
    updtrec(nboxes,level(nboxes),f(1,:));
    if prt > 1,
      updtoptl(i,u(i),x0(i,1),iopt,s+1,f0(1));
    end
  end
  for j=1:L(i)-1
    nchild = nchild + 1;
    splval = split1(x0(i,j),x0(i,j+1),f0(j),f0(j+1));
    if f0(j) <= f0(j+1) | s + 2 < smax
      nboxes = nboxes + 1;
      if f0(j) <= f0(j+1)
        level0 = s + 1;
      else
        level0 = s + 2;
      end
      [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] =
genbox(par,level0,-nchild,f0(j));
      updtrec(nboxes,level(nboxes),f(1,:));
      if prt > 1,
        updtoptl(i,x0(i,j),splval,iopt,level0,f0(j));
      end
    else
      x(i) = x0(i,j);
      nbasket = nbasket + 1;
      xmin(:,nbasket) = x;
      fmi(nbasket) = f0(j);
      if prt > 1,
        updtoptl(i,x0(i,j),splval,iopt,smax,f0(j));
      end
    end
    nchild = nchild + 1;
    if f0(j+1) < f0(j) | s + 2 < smax
      nboxes = nboxes + 1;
      if f0(j+1) < f0(j)
        level0 = s + 1;
      else
        level0 = s + 2;
      end
      [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] =
genbox(par,level0,-nchild,f0(j+1));
      updtrec(nboxes,level(nboxes),f(1,:));
      if prt > 1,
        updtoptl(i,splval,x0(i,j+1),iopt,level0,f0(j+1));
      end
    else
      x(i) = x0(i,j+1);
      nbasket = nbasket + 1;
      xmin(:,nbasket) = x;
      fmi(nbasket) = f0(j+1);
      if prt > 1,
        updtoptl(i,splval,x0(i,j+1),iopt,smax,f0(j+1));
      end
    end
  end
  if x0(i,L(i)) < v(i) % in that case the box at the boundary gets level s + 1
    nchild = nchild + 1;
    nboxes = nboxes + 1;
    [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] = genbox(par,s+1,-
nchild,f0(L(i)));
    updtrec(nboxes,level(nboxes),f(1,:));
    if prt > 1,
      updtoptl(i,x0(i,L(i)),v(i),iopt,s+1,f0(L(i)));
    end
  end
else
  if prt > 1 & u(i) < x0(i,1)
    updtoptl(i,u(i),x0(i,1),iopt,smax,f0(1));
  end
  for j=1:L(i)
    x(i) = x0(i,j);
    nbasket = nbasket + 1;
    xmin(:,nbasket) = x;
    fmi(nbasket) = f0(j);
    if prt > 1 & j < L(i)
      splval = split1(x0(i,j),x0(i,j),f0(j),f0(j+1));
      updtoptl(i,x0(i,j),splval,iopt,smax,f0(j));
      updtoptl(i,splval,x0(i,j+1),iopt,smax,f0(j+1));
    end
    if prt > 1 & x0(i,L(i)) < v(i)
      updtoptl(i,x0(i,L(i)),v(i),iopt,smax,f0(L(i)));
    end
  end
end
```

SPLIT FUNCTION:

The Split function splits box # par at level s in its ith coordinate into two or three children and inserts its children and their parameters in the list.

```
function [xbest,fbest,xmin,fmi,ipar,level,ichild,f,flag,ncall] =
split(fcn,data,i,s,smax,par,n0,u,v,x,y,x1,x2,z,xmin,fmi,ipar,level,ichild,f,xbes
t,fbest,stop,prt)

global nbasket nboxes nglob nsweep nsweepbest record xglob
ncall = 0;
n = length(x);
iopt = [];
if prt > 1
  [w1,w2] =  bounds(n,n0,x,y,u,v);
  for iglob = 1:nglob
    if w1 <= xglob(:,iglob) & xglob(:,iglob) <= w2
      iopt = [iopt, iglob];
    end
  end
end
flag = 1;
x(i) = z(2);
f(2,par) = feval(fcn,data,x);
```

```
ncall = ncall + 1;
if f(2,par) < fbest
  fbest = f(2,par);
  xbest = x;
  nsweepbest = nsweep;
  if stop(1) > 0 & stop(1) < 1
    flag = chrelerr(fbest,stop);
  elseif stop(1) == 0
    flag = chvtr(fbest,stop(2));
  end
  if ~flag,return,end
end
splval = split1(z(1),z(2),f(1,par),f(2,par));
if s + 1 < smax
  if f(1,par) <= f(2,par)
    nboxes = nboxes + 1;
    [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] =
genbox(par,s+1,1,f(1,par));
    updtrec(nboxes,level(nboxes),f(1,:));
    if prt > 1,
      updtoptl(i,z(1),splval,iopt,s+1,f(1,par));
    end
    if s + 2 < smax
      nboxes = nboxes + 1;
      [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] =
genbox(par,s+2,2,f(2,par));
      updtrec(nboxes,level(nboxes),f(1,:));
    else
      x(i) = z(2);
      nbasket = nbasket + 1;
      xmin(:,nbasket) = x;
      fmi(nbasket) = f(2,par);
    end
    if prt > 1,
      updtoptl(i,splval,z(2),iopt,s+2,f(2,par));
    end
  else
    if s + 2 < smax
      nboxes = nboxes + 1;
      [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] =
genbox(par,s+2,1,f(1,par));
      updtrec(nboxes,level(nboxes),f(1,:));
    else
      x(i) = z(1);
      nbasket = nbasket + 1;
      xmin(:,nbasket) = x;
      fmi(nbasket) = f(1,par);
    end
    if prt > 1,
      updtoptl(i,z(1),splval,iopt,s+2,f(1,par));
    end
    nboxes = nboxes + 1;
    [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] =
genbox(par,s+1,2,f(2,par));
    updtrec(nboxes,level(nboxes),f(1,:));
    if prt > 1,
      updtoptl(i,splval,z(2),iopt,s+1,f(2,par));
    end
  end
  if z(2) ~= y(i)
    if abs(z(2)-y(i)) > abs(z(2)-z(1))*(3-sqrt(5))*0.5
      nboxes = nboxes + 1;
      [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] =
genbox(par,s+1,3,f(2,par));
      updtrec(nboxes,level(nboxes),f(1,:));
      if prt > 1,
        updtoptl(i,z(2),y(i),iopt,s+1,f(2,par));
      end
    else
      if s + 2 < smax
        nboxes = nboxes + 1;
        [ipar(nboxes),level(nboxes),ichild(nboxes),f(1,nboxes)] =
genbox(par,s+2,3,f(2,par));
        updtrec(nboxes,level(nboxes),f(1,:));
      else
        x(i) = z(2);
        nbasket = nbasket + 1;
        xmin(:,nbasket) = x;
        fmi(nbasket) = f(2,par);
      end
      if prt > 1,
        updtoptl(i,z(2),y(i),iopt,s+2,f(2,par));
      end
    end
  end
else
  x(i) = z(1);
  nbasket = nbasket + 1;
  xmin(:,nbasket) = x;
  fmi(nbasket) = f(1,par);
  x(i) = z(2);
  nbasket = nbasket + 1;
  xmin(:,nbasket) = x;
  fmi(nbasket) = f(2,par);
  if prt > 1,
    updtoptl(i,z(1),splval,iopt,smax,f(1,par));
    updtoptl(i,splval,z(2),iopt,smax,f(2,par));
    if z(2) ~= y(i)
      updtoptl(i,z(2),y(i),iopt,smax,f(2,par));
    end
  end
end
```

BOUNDS FUNCTION:

The Bounds function computes the bounds for a box with base vertex x and opposite vertex y.

```
function [w1,w2] = bounds(n,n0,x,y,u,v)

w1 = zeros(n,1);
w2 = w1;
for i = 1:n
  if n0(i) == 0
    w1(i) = u(i);
    w2(i) = v(i);
  else
    w1(i) = min(x(i),y(i));
    w2(i) = max(x(i),y(i));
  end
end
```

LSEARCH FUNCTION:

```
function [xmin,fmi,ncall,flag] =
lsearch(fcn,data,x,f,f0,u,v,nf,stop,maxstep,gamma,hess)

global nsweep nsweepbest
ncall = 0;
n = length(x);
x0 = min(max(u,0),v); % absolutely smallest point
if nargin < 10, maxstep = 50; end
if nargin < 11, gamma = eps; end
if nargin < 12, hess = ones(n,n); end
flag = 1;
eps0 = 0.001;
nloc = 1;
small = 0.1;
smaxls = 15;
[xmin,fmi,g,G,nfcsearch] = csearch(fcn,data,x,f,u,v,hess);
xmin = max(u,min(xmin,v));
ncall = ncall + nfcsearch;
xold = xmin;
fold = fmi;
if stop(1) > 0 & stop(1) < 1
  flag = chrelerr(fmi,stop);
elseif stop(1) == 0
  flag = chvtr(fmi,stop(2));
end
if ~flag,return,end
d = min(min(xmin-u,v-xmin),0.25*(1+abs(x-x0)));
p = minq(fmi,g,G,-d,d,0);
if norm(p),
  x = xmin + p;
  x = max(u,min(x,v));
  f1 = feval(fcn,data,x);
  ncall = ncall + 1;
  alist = [0 1];
  flist = [fmi f1];
  fpred = fmi + g'*p + 0.5*p'*G*p;
  [alist,flist,nfls] = gls(fcn,data,u,v,xmin,p,alist,flist,nloc,small,smaxls);
  ncall = ncall + nfls;
  [fminew,i] = min(flist);
  if fminew == fmi
    i = find(~alist);
  else
    fmi = fminew;
  end
  xmin = xmin + alist(i)*p;
  xmin = max(u,min(xmin,v));
  gain = f - fmi;
  if stop(1) > 0 & stop(1) < 1
    flag = chrelerr(fmi,stop);
  elseif stop(1) == 0
    flag = chvtr(fmi,stop(2));
  end
  if ~flag,return,end
  if fold == fmi
    r = 0;
  elseif fold == fpred
    r = 0.5;
  else
    r = (fold-fmi)/(fold-fpred);
  end
else
```

```
  gain = f - fmi;
  r = 0;
end
diag = 0;
ind = find(u < xmin & xmin < v);
b = abs(g)'*max(abs(xmin),abs(xold));
nstep = 0;
while ncall < nf & nstep < maxstep & (diag | length(ind) < n | (stop(1) == 0 &
fmi - gain <= stop(2)) | (b >= gamma*(f0-f) & gain > 0))
  nstep = nstep + 1;
  delta = abs(xmin)*eps^(1/3);
  j = find(~delta);
  if ~isempty(j)
    delta(j) = eps^(1/3)*ones(size(j));
  end
  [x1,x2] = neighbor(xmin,delta,u,v);
  f = fmi;
  if length(ind) < n & (b < gamma*(f0-f) | ~gain)
    ind1 = find(xmin == u | xmin == v);
    for k=1:length(ind1)
      i = ind1(k);
      x = xmin;
      if xmin(i) == u(i)
        x(i) = x2(i);
      else
        x(i) = x1(i);
      end
      f1 = feval(fcn,data,x);
      ncall = ncall + 1;
      if f1 < fmi
        alist = [0 x(i)-xmin(i)];
        flist = [fmi f1];
        p = zeros(n,1);
        p(i) = 1;
        [alist,flist,nfls] = gls(fcn,data,u,v,xmin,p,alist,flist,nloc,small,6);
        ncall = ncall + nfls;
        [fminew,j] = min(flist);
        if fminew == fmi
          j = find(~alist);
        else
          fmi = fminew;
        end
        xmin(i) = xmin(i) + alist(j);
      else
        ind1(k) = 0;
      end
    end
    xmin = max(u,min(xmin,v));
    if ~sum(ind1),break,end
    delta = abs(xmin)*eps^(1/3);
    j = find(~delta);
    if ~isempty(j)
      delta(j) = eps^(1/3)*ones(size(j));
    end
    [x1,x2] = neighbor(xmin,delta,u,v);
  end
  if abs(r-1) > 0.25 | ~gain | b < gamma*(f0-f)
    [xmin,fmi,g,G,x1,x2,nftriple] = triple(fcn,data,xmin,fmi,x1,x2,u,v,hess);
    ncall = ncall + nftriple;
    diag = 0;
  else
    [xmin,fmi,g,G,x1,x2,nftriple] = triple(fcn,data,xmin,fmi,x1,x2,u,v,hess,G);
    ncall = ncall + nftriple;
    diag = 1;
  end
  xold = xmin;
  fold = fmi;
  if stop(1) > 0 & stop(1) < 1
    flag = chrelerr(fmi,stop);
  elseif stop(1) == 0
    flag = chvtr(fmi,stop(2));
  end
  if ~flag,return,end
  if r < 0.25
    d = 0.5*d;
  elseif r > 0.75
    d = 2*d;
  end
  p = minq(fmi,g,G,max(-d,u-xmin),min(d,v-xmin),0);
  if ~norm(p) & ~diag & length(ind) == n,break,end
  if norm(p),
    fpred = fmi + g'*p + 0.5*p'*G*p;
    x = xmin + p;
    f1 = feval(fcn,data,x);
    ncall = ncall + 1;
    alist = [0 1];
    flist = [fmi f1];
    [alist,flist,nfls] = gls(fcn,data,u,v,xmin,p,alist,flist,nloc,small,smaxls);
    ncall = ncall + nfls;
    [fmi,i] = min(flist);
    xmin = xmin + alist(i)*p;
    xmin = max(u,min(xmin,v));
    if stop(1) > 0 & stop(1) < 1
      flag = chrelerr(fmi,stop);
    elseif stop(1) == 0
      flag = chvtr(fmi,stop(2));
    end
    if ~flag,return,end
    gain = f - fmi;
    if fold == fmi
      r = 0;
    elseif fold == fpred
      r = 0.5;
    else
      r = (fold-fmi)/(fold-fpred);
    end
    if fmi < fold
      fac = abs(1-1/r);
      eps0 = max(eps,min(fac*eps0,0.001));
    else
      eps0 = 0.001;
    end
  else
    gain = f - fmi;
    if ~gain
      eps0 = 0.001;
      fac = Inf;
      r = 0;
    end
  end
  ind = find(u < xmin & xmin < v);
  b = abs(g)'*max(abs(xmin),abs(xold));
end
```

BASKET 1 FUNCTION:

The Basket 1 function checks whether a candidate for the 'shopping basket' lies in the 'domain of attraction' of a point already in the 'shopping basket'.

```
function [xbest,fbest,xmin,fmi,loc,flag,ncall] = basket1(fcn,data,x,f
xmin,fmi,xbest,fbest,stop,nbasket)

global nsweep nsweepbest
loc = 1;
flag = 1;
ncall = 0;
if ~nbasket, return, end
for k = 1:nbasket
  dist(k) = norm(x - xmin(:,k));
end
[dist1,ind] = sort(dist);
for k = 1:nbasket
  i = ind(k);
  p = xmin(:,i) - x;
  y1 = x + 1/3*p;
  f1 = feval(fcn,data,y1);
  ncall = ncall + 1;
  if f1 <= max(fmi(i),f)
    y2 = x + 2/3*p;
    f2 = feval(fcn,data,y2);
    ncall = ncall + 1;
    if f2 <= max(f1,fmi(i))
      if f < min(min(f1,f2),fmi(i))
        fmi(i) = f;
        xmin(:,i) = x;
        if fmi(i) < fbest
          fbest = fmi(i);
          xbest = xmin(:,i);
          nsweepbest = nsweep;
          if stop(1) > 0 & stop(1) < 1
            flag = chrelerr(fbest,stop);
          elseif stop(1) == 0
            flag = chvtr(fbest,stop(2));
          end
          if ~flag,return,end
        end
        loc = 0;break
      elseif f1 < min(min(f,f2),fmi(i))
        fmi(i) = f1;
        xmin(:,i) = y1;
        if fmi(i) < fbest
          fbest = fmi(i);
          xbest = xmin(:,i);
          nsweepbest = nsweep;
          if stop(1) > 0 & stop(1) < 1
            flag = chrelerr(fbest,stop);
          elseif stop(1) == 0
            flag = chvtr(fbest,stop(2));
          end
          if ~flag,return,end
        end
        loc = 0;break
      elseif f2 < min(min(f,f1),fmi(i))
        fmi(i) = f2;
        xmin(:,i) = y2;
        if fmi(i) < fbest
          fbest = fmi(i);
          xbest = xmin(:,i);
          nsweepbest = nsweep;
          if stop(1) > 0 & stop(1) < 1
            flag = chrelerr(fbest,stop);
          elseif stop(1) == 0
            flag = chvtr(fbest,stop(2));
          end
          if ~flag,return,end
        end
        loc = 0;break
      else
        loc = 0;break
      end
    end
  end
end
```

HESSIAN:

The Hessian computes the element G(i,k) of the Hessian of the local quadratic model.

```
function h = hessian(i,k,x,x0,f,f0,g,G)

h = f-f0-g(i)*(x(i)-x0(i))-g(k)*(x(k)-x0(k))-0.5*G(i,i)*(x(i)-x0(i))^2-
0.5*G(k,k)*(x(k)-x0(k))^2;
h = h/(x(i)-x0(i))/(x(k)-x0(k));
```

GLS FUNCTION:

The GLS function provides local and global line search for noiseless functions.

```
 function
[alist,flist,nf]=gls(func,data,xl,xu,x,p,alist,flist,nloc,small,smax,prt)

if nargin==1,
  disp(' ');
  disp('******** debug mode; old data are used ********')
  disp(' ');
  prt1=func;
  load glsinput;
  prt=prt1;
else
  if nargin<9, nloc=1; end;
  if nargin<10, small=0.1; end;
  if nargin<11, smax=10; end;
  if nargin<12, prt=0; end;
  save glsinput;
end;

if prt>1,
  disp('********************* gls ********************');
  format short;
end;

sinit=size(alist,2

% get 5 starting points (needed for lslocal)
bend=0;lsrange;
lsinit;
```

```
nf=s-sinit;

while s<min(5,smax),
  if nloc==1,
    lspar;
    if s>3 & monotone & (abest==amin | abest==amax),
      if prt>1, disp('return since monotone'); end;
      nf=s-sinit
lsdraw; return;
    end;
  else
    lsnew;
  end;
end;

saturated=0;
if nmin==1,
  if monotone & (abest==amin | abest==amax),
    if prt>1, disp('return since monotone');
 end;
    nf=s-sinit;
    lsdraw; return;
  end;
  if s==5, lsquart;
end;

  lsdescent;
  lsconvex;
  if convex,
    if prt>1, disp('return since convex'); end;
    nf=s-sinit;
    lsdraw; return;
  end;
end;
sold=0;
while 1,
  lsdraw;
  if prt>1, disp('***** new refinement iteration *****'); end;
  lsdescent;
  lssat;
  if saturated | s==sold | s>=smax,
    if saturated & prt>1, disp('return since saturated'); end;
    if s==sold   & prt>1, disp('return since s==sold');   end;
    if s>=smax   & prt>1, disp('return since s>=smax');   end;
    lsdraw;break;
  end;
  sold=s; nminold=nmin; if prt>1, nmin, end;
  if ~saturated & nloc>1,
    lssep;
  end;
  lslocal;
  if nmin>nminold, saturated=0; end;
end;

nf=s-sinit;
```

ELS FUNCTION:

The ELS function provides new linear line search using only initial gradient.

```
function [alp,x,f,eff,nf]=els(case,x0,f0,p,gTp,fid);
```

```
if ~exist('fid'), fid=0; end;
if fid>0, fprintf(fid,'*********** new line search **********\n');
 end;

if gTp>=0,
  eff=-1;
  x=x0;f=f0;
  return;
end;

maxit=10;
ftol=.001;
xtol=.1;
stpmin = 0;
stpmax = 1e20;
 [memory,ier]=init(f0,gTp,ftol,xtol,stpmin,stpmax,fid);
alp=1;
nf=0;
task='VALUE';

while 1,
  ier=0;
  if task(1)=='V',
    x=x0+alp*p;
    f=func1(case,x);
    nf=nf+1;
    eff=0;
  elseif task(1)=='C',
    if fid>0, fprintf(fid,task);
end;
    eff=1;
  elseif task(1)=='W',
    if fid>0, fprintf(fid,task);
end;
    eff=2;
  else % task(1)=='E',
    if fid>0, fprintf(fid,task);
end;
    eff=0;
    x=x0;f=f0;
    ier=1;
  end;
  if nf>=maxit | ier>0 | eff>0, break;
end;
    [alp,task,memory]=step(alp,f,memory);
end;

if f>f0,x=x0;f=f0;
end;

if fid>0 & nf>0,
  format='eff=%1.0f f=%16.8e alp=%10.4f mu=%8.1f nf=%2.0f%4s\n';
  mu=(f-f0)/(alp*gTp);
  if length(task)<12, task='VALUE    max';
end;
  fprintf(fid,format,eff,f,alp,mu,nf,task(9:12));
end;
```

MINQ FUNCTION:

The Minq function minimizes an affine quadratic form subject to simple bounds, using coordinate searches and reduced subspace minimizations using LDL^T factorization updates.

```
 function [x,fct,ier,nsub]=minq(gam,c,G,xu,xo,prt,xx);

if prt>0, printlevel=prt, end;
convex=0;
n=size(G,1);
maxit=3*nnitrefmax=3;

if nargin<7,
  xx=zeros(n,1);
end;
xx=max(xu,min(xx,xo));

hpeps=100*eps;
G=G+spdiags(hpeps*diag(G),0,n,n);

K=logical(zeros(n,1));  % initially no rows in factorization
if issparse(G), L=speye(n); else L=eye(n); end;
dd=ones(n,1);

free=logical(zeros(n,1));
nfree=0;
nfree_old=-1;

fct=inf;
nsub=0;
unfix=1;
nitref=0;
improvement=1;

while 1,
  if prt>1, disp('enter main loop'); end;
  if norm(xx,inf)==inf, error('infinite xx in minq.m'); end;
  g=G*xx+c;
  fctnew=gam+0.5*xx'*(c+g);
  if ~improvement,
    if prt,
      disp('terminate: no improvement in coordinate search');
    end;
    ier=0; break;
  elseif nitref>nitrefmax,
    if prt, disp('terminate: nitref>nitrefmax'); end;
    ier=0; break;
  elseif nitref>0 & nfree_old==nfree & fctnew >= fct,
    if prt,
      disp('terminate: nitref>0 & nfree_old==nfree & fctnew>=fct');
    end;
    ier=0; break;
  elseif nitref==0,
    x=xx;
    fct=min(fct,fctnew);
    if prt>1, fct, end;
    if prt>2, X=x', fct, end;
  else
    x=xx;
    fct=fctnew;
    if prt>1, fct, end;
    if prt>2, X=x', fct, end;
  end;
```

```
if nitref==0 & nsub==maxit,
  if prt,
    disp('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
    disp('!!!!!           minq          !!!!!');
    disp('!!!!! incomplete minimization !!!!!');
    disp('!!!!!   too many iterations   !!!!!');
    disp('!!!!!     increase maxit      !!!!!');
    disp('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
  else
    disp('iteration limit exceeded');
  end;
  ier=99;
  break;
end;

count=0;    k=0;
while 1,
  while count<=n,
    count=count+1;
    if k==n, k=0; end;
    k=k+1;
    if free(k) | unfix, break; end;
  end;
  if count>n,
  end;
  q=G(:,k);
  alpu=xu(k)-x(k); alpo=xo(k)-x(k); % bounds on step

  [alp,lba,uba,ier]=getalp(alpu,alpo,g(k),q(k));
  if ier,
    x=zeros(n,1);
    if lba, x(k)=-1; else x(k)=1; end;
    if prt,
      gTp=g(k),pTGp=q(k),quot=pTGp/(norm(p,1)^2*norm(G(:),inf))
      disp('minq: function unbounded below in coordinate direction');
      disp('      unbounded direction returned');
      disp('      possibly caused by roundoff');
    end;
    if prt>1,
      disp('f(alp*x)=gam+gam1*alp+gam2*alp^2/2, where');
      gam1=c'*x
      gam2=x'*(G*x)
      ddd=diag(G);
      min_diag_G=min(ddd)
      max_diag_G=max(ddd)
    end;
    return;
  end;
  xnew=x(k)+alp;
  if prt & nitref>0,
    xnew,alp
  end;

  if lba | xnew<=xu(k),
    if prt>2, disp([num2str(k), ' at lower bound']); end;
    if alpu~=0,
      x(k)=xu(k);
      g=g+alpu*q;
      count=0;
    end;
    free(k)=0;
  elseif uba | xnew>=xo(k),
    if prt>2, disp([num2str(k), ' at upper bound']); end;
```

```
      if alpo~=0,
        x(k)=xo(k);
        g=g+alpo*q;
        count=0;
      end;
      free(k)=0;
    else
      if prt>2, disp([num2str(k), ' free']); end;
      if alp~=0,
        if prt>1 & ~free(k),
          unfixstep=[x(k),alp],
        end;
        x(k)=xnew;
        g=g+alp*q;
        free(k)=1;
      end;
    end;

  end;

  nfree=sum(free);
  if (unfix & nfree_old==nfree),
    g=G*x+c;
    nitref=nitref+1;
    if prt>0,
      disp('optimum found; iterative refinement tried');
    end;
  else
    nitref=0;
  end;
  nfree_old=nfree;
  gain_cs=fct-gam-0.5*x'*(c+g);
  improvement=(gain_cs>0 | ~unfix);

  if prt,
    nfree=pr01('csrch ',free);
  end;
  if prt, gain_cs, end;
  if prt>2, X=x', end;

  xx=x;
  if ~improvement | nitref>nitrefmax,
  elseif nitref>nitrefmax,
  elseif nfree==0,
    if prt>0,
      disp('no free variables - no subspace step taken')
    end;
    unfix=1;
  else
    minqsub;
    if ier, return; end;
  end;

  if prt>0,
    nfree=pr01('ssrch ',free);
    disp(' ');
    if unfix & sum(nfree)<n,
      disp('bounds may be freed in next csearch');
    end;
  end;


end;
```

```
if prt>0,
  fct
  disp('################## end of minq ###################');
end;
```