Flow regime prediction for the Merri Creek Catchment

Matthew Burns, Nick Bond, and Tim Fletcher

November 7, 2014

1 Introduction

Future urban development in the upper reaches of the Merri Creek catchment could substantially impact flow regimes as well as water quality. To assess these impacts, Melbourne Water—the regional waterway management authority—has contracted external consultants to develop MUSIC models that will provide an indication of future urban flow regimes. However, in order to assess the impacts of future land-use changes on the hydrology of Merri Creek, information on natural flow regimes is needed. This requires a baseline time series that estimates the mean daily flow in the absence of urban development. These data are required for two reaches of Merri Creek. In this work, we use rainfall-runoff modelling to make such predictions.

2 Methods

Our aim was to derive long-term time series of mean daily flow for two reaches of Merri Creek, assuming pre-urban land-use (Figure 1). The upstream reach (1), has a catchment area of $\sim$54 km^2 and some urban land-use (total imperviousness $\leq$5%). The more downstream reach (2) has a larger catchment ($\sim$271 km^2), but similar land-use. The upstream reach is not gauged. While there are several gauges near the downstream reach, the flows recorded represent a period of current land-use. The approach we have taken to predict flows is rainfall-runoff modelling and the steps included:

1. Identify gauged undeveloped catchments in the region with similar environmental attributes to the two Merri Creek reaches.
2. For the identified catchments, build and calibrate rainfall-runoff models.
3. Transpose optimized rainfall-runoff model parameters to the Merri Creek reaches.
4. Simulate flow time-series at the relevant reaches.

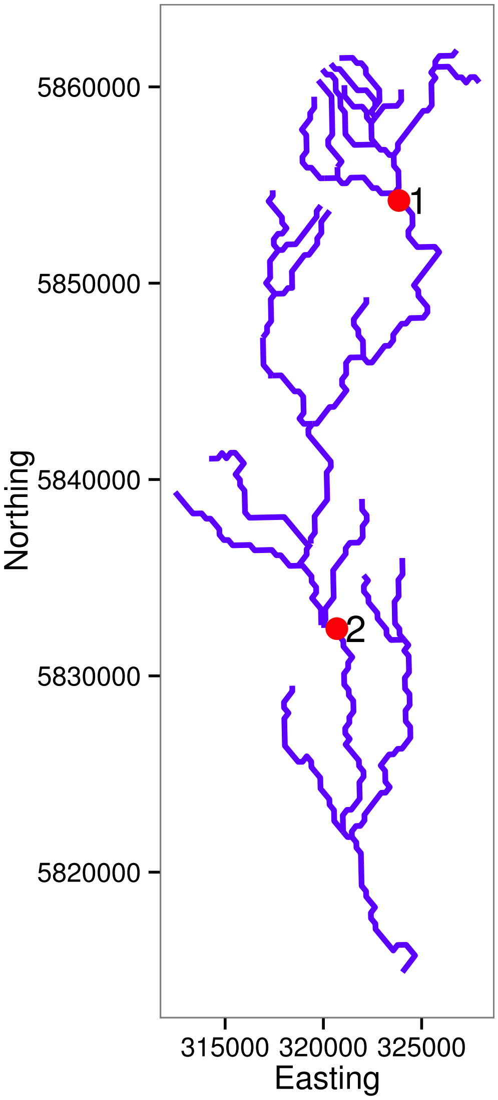

Figure 1: The Merri Creek catchment. Red circles = sites (or reaches) in the catchment where flow predictions were required.

2.1 Catchment identification

Appropriate reference gauges for model calibration were selected based on a quantitate evaluation of catchment similarity. We extracted a suite of variables describing catchment climate and physiography for all of the catchments gauged by Melbourne Water as well as for the two focus reaches. These variables were each then range standardized to a common scale (0-1), and combined in calculating the Euclidean distance between the focus reaches and each of the gauged catchments (identical catchments would have a distance of zero). We then ranked gauges based on their overall catchment similarity (as defined by the distance metric), and selected the most similar catchments (after filtering out those with high levels of urbanization) for the calibration process. The most similar gauged undeveloped catchment to the upstream reach was 230119A (Deep Creek at Lancefield). For the downstream reach, two such catchments were identified—230107A (Deep Creek at Konagaderra) and 230211A (Bolinda Creek at Lancefield Road). We selected the Bolinda Creek gauge for this work because its catchment area is more similar to the downstream reach compared to 230107A. We then aimed to calibrate rainfall-runoff models for 230119A and 230211A.

2.2 Rainfall-runoff modelling

Rainfall-runoff modelling was carried out using a modified version of SIMHYD (Peel et al., 2000). Testing showed that the original SIMHYD model had difficulty in simulating dry systems such as Merri Creek. We thus modified SIMHYD by including a store which represented bank storage, and this greatly improved the model's ability to simulate dry systems. We then sourced relevant input data for the modelling—rainfall, evapotranspiration, and streamflow. For rainfall, we used the gridded daily data from the Australian Bureau of Meteorology. Another gridded dataset was used for evapotranspiration. Streamflow data was sourced from Melbourne Water and included the gauged records at 230119A (Deep Creek at Lancefield) and 230211A (Bolinda Creek at Lancefield Road). Any poor quality streamflow data was set to missing.

We then aimed to calibrate modified SIMHYD models for both Deep and Bolinda creeks. To calibrate the models, we used Generalized Likelihood Uncertainty Estimation (GLUE; Beven and Freer, 2001). In this approach, the model is run numerous times using a large parameter space (shown below). Model runs which satisfy an objective function are retained (1). Using the output from these runs, the reported simulated flow is a range with lower and upper bounds.

Model parameters for the modified version of SIMHYD are listed below. The range of parameter values we considered are shown in parentheses.

- insc—interception store capacity (0.5-5 mm).
- coeff—maximum infiltration loss (20-400 mm).

- sq—infiltration loss exponent (0-10).
- smsc—soil moisture store capacity (20-500 mm).
- sub—constant of proportionality in interflow equation (0-1).
- crak—constant of proportionality in groundwater recharge equation (0-1).
- k—baseflow linear recession parameter (0.003-0.5).
- psize—size of bank storage store as a fraction of catchment area (0-0.3).
- carea—catchment area (km^2).

$$E = 1 - \frac{\sum_{i=1}^{n} (O_i^\lambda - P_i^\lambda)^2}{\sum_{i=1}^{n} (O_i^\lambda - \bar{O}^\lambda)^2} \tag{1}$$

where E is the Nash-Sutcliffe model efficiency coefficient (Nash and Sutcliffe, 1970), O is observed flow and P is simulated flow. λ transforms the data and for this work was set to 0.5 (meaning that low-flows were the focus of model calibration). Transforming the data in this way reduces the influence of high-flows in the calcuation of E (Krause et al., 2005).

Calibration of Deep Creek

We devised 5000 different model runs for Deep Creek by randomly sampling the parameter space. The modelling period spanned the record of available streamflow data (2000/03/01 to 2013/09/30; YYYY/MM/DD). We ran the models and each time assessed the quality of calibration using (1). Models were retained when E was $\geq$0.60—this signifies a satisfactory model fit (Chiew and McMahon, 1993). We found only 3 acceptable models for Deep Creek and thus selected the best one (Table 1). Graphical plots show good agreement between simulated flows (from the best model) and the observed data (Figures 2 and 3). Note that because few acceptable models were found, we could not apply the full GLUE methodology. This meant that uncertainty bounds could not be placed on the simulated flow.

Calibration of Bolinda Creek

It was more difficult to calibrate a rainfall-runoff model for Bolinda Creek than that for Deep Creek. Several thousand model runs using the same parameter space as above yielded no acceptable models (E <0.60). We subsequently investigated parameter values that resulted in very poor model fits and restricted the parameter ranges accordingly. For example, all model runs with sub >0.20 were of poor quality, so we restricted the parameter range of sub from 0$\leq$sub$\leq$1 to 0$\leq$sub$\leq$0.2. We then ran numerous models (using the restricted parameter space) and in doing so, found 3 acceptable models for Bolinda Creek. Like Deep

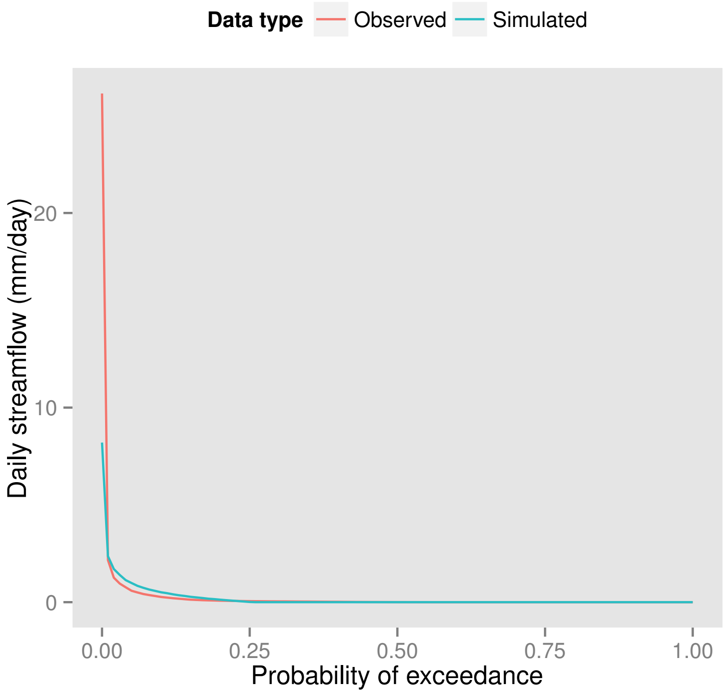

Figure 2: Flow duration curves for Deep Creek.

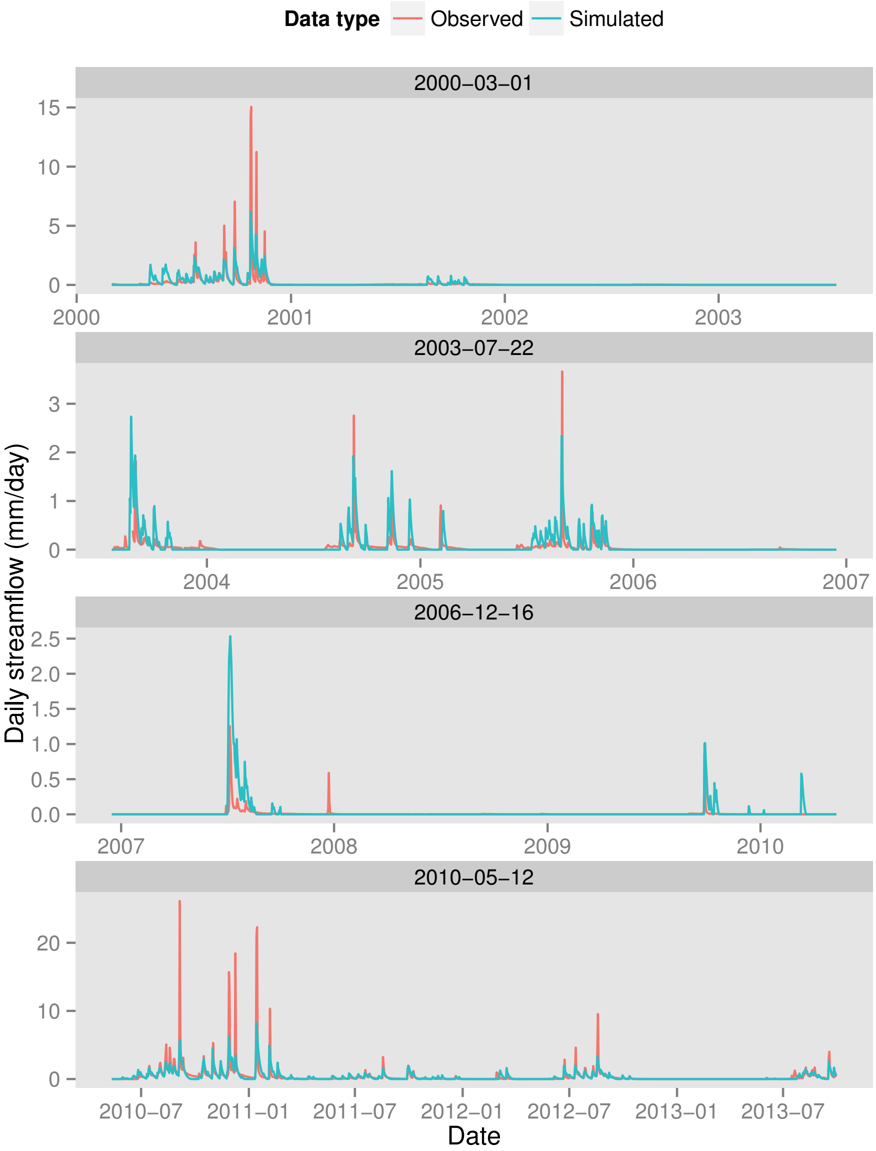

Figure 3: Flow time series for Deep Creek.

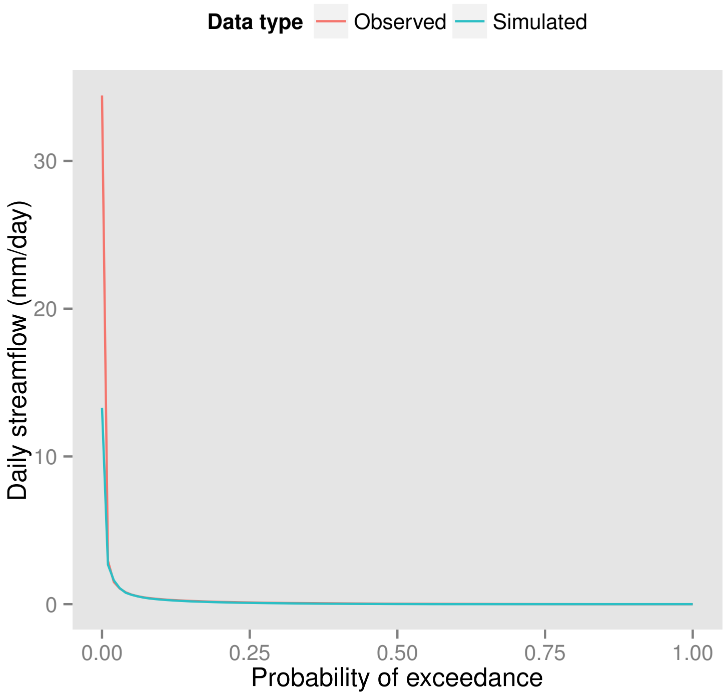

Figure 4: Flow duration curves for Bolinda Creek.

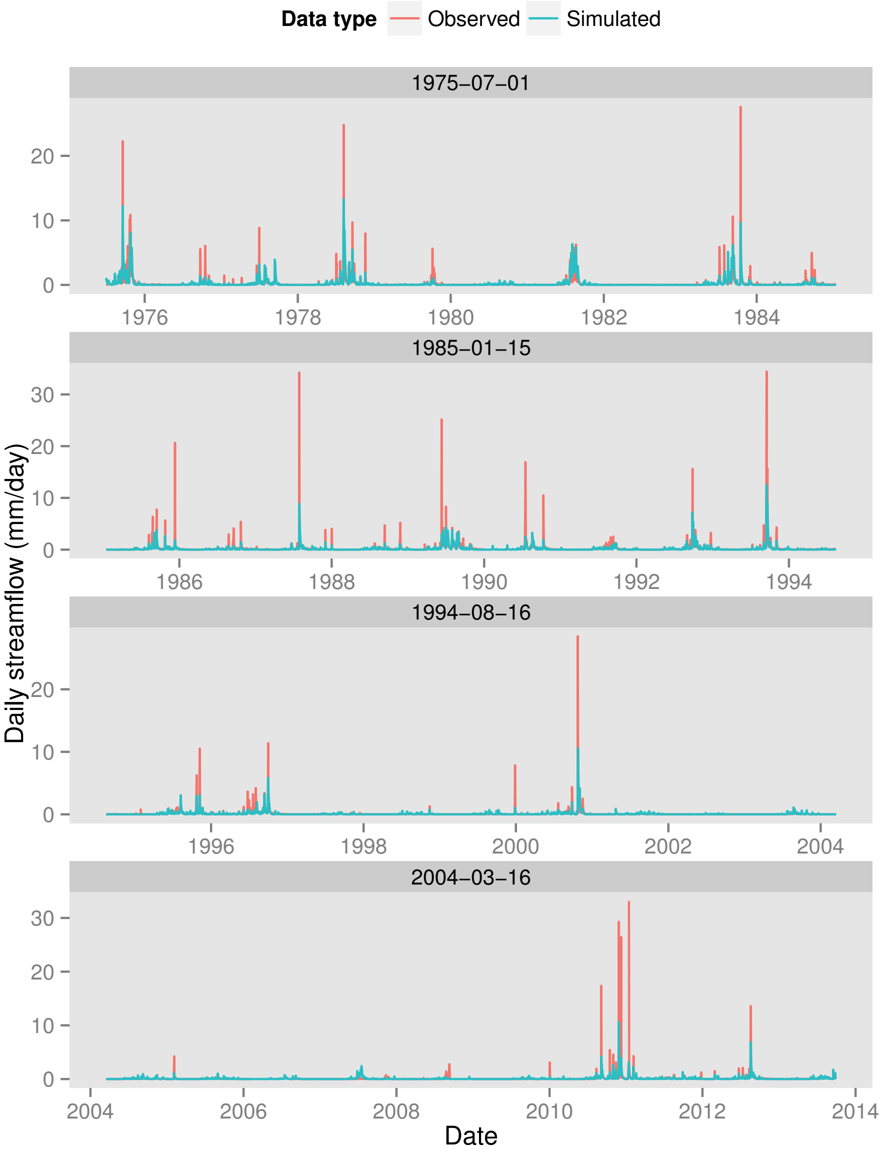

Figure 5: Flow time series for Bolinda Creek.

Table 1: Parameter values for the best rainfall-runoff models.

Parameter	Deep Creek	Bolinda Creek
insc	2.118	3.681
coeff	367.885	249.165
sq	0.507	0.675
smsc	431.807	209.049
sub	0.027	0.031
crak	0.725	0.126
k	0.194	0.256
psize	0.051	0.002
carea	217.381	95.331

Creek, the best of these 3 models was selected (*E*=0.62; parameters shown in Table 1). Inspection of flow duration curves and flow time series show good agreement between simulated flow and the observed data (Figures 4 and 5).

2.3 Parameter transposition

Rainfall-runoff model parameters for the upstream Merri Creek reach were simply the parameters from the best Deep Creep model (see Table 1), but with carea = 54 km^2. Similarly, parameters for the downstream focus reach were those from the best Bolinda Creek model (and with carea = 271 km^2).

Table 2: Characteristics of the study catchments.

	Merri Creek (upper reach)	Deep Creek	Merri Creek (lower reach)	Bolinda Creek
Catchment area (km^2)	54	217	271	95

2.4 Flow simulation

We derived long-term series of flow for the two focus reaches by running the modified version of SIMHYD using 1) the transposed model parameters, and 2) long-term input data (gridded rainfall and evapotranspiration; 1900 to 2013).

3 Results and validation

Predicted flows for the upper and lower focus reaches of Merri Creek are shown in Figures 6 and 7 respectively.

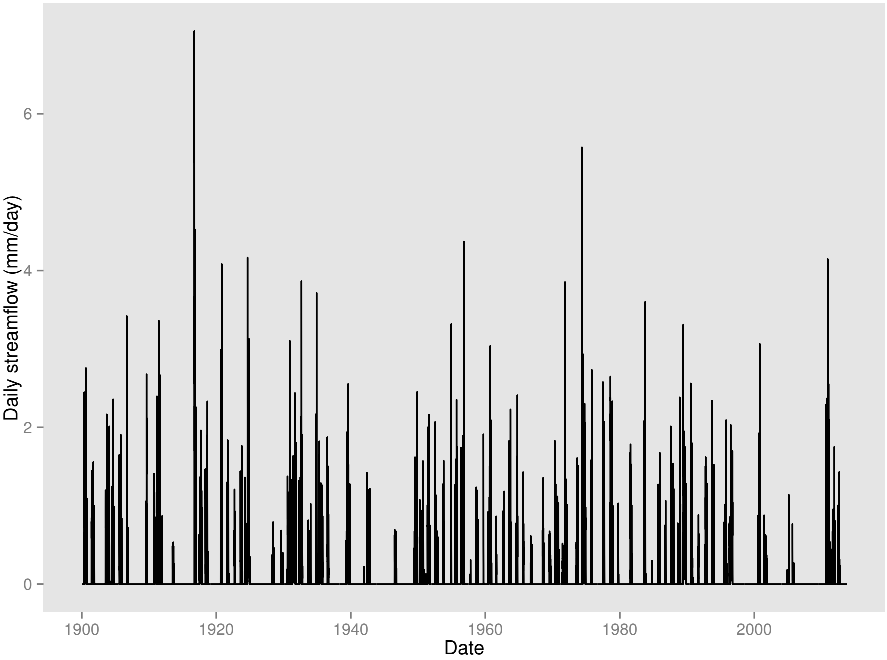

Figure 6: Predicted flow time series for the upper reach of Merri Creek.

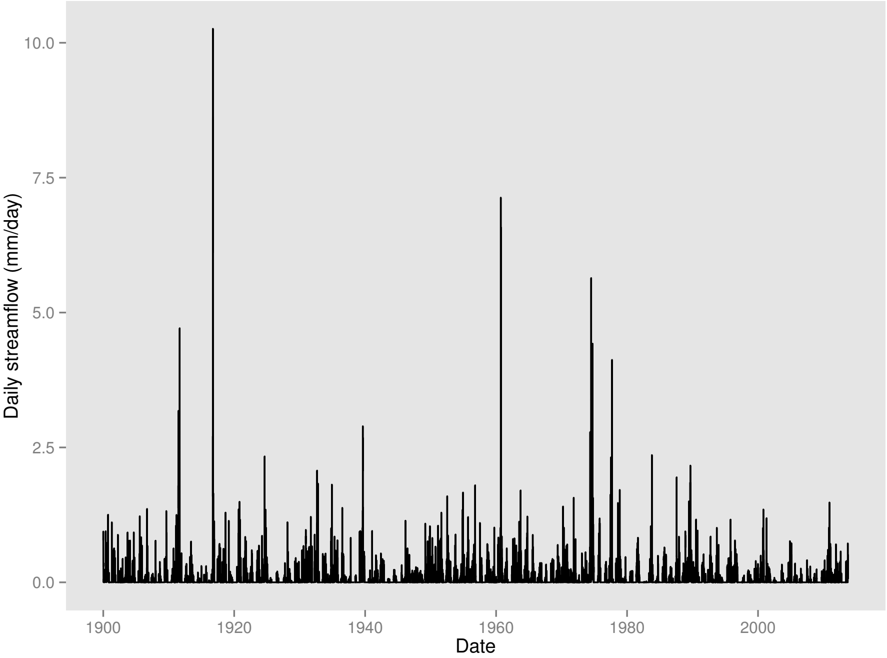

Figure 7: Predicted flow time series for the downstream reach of Merri Creek.

References

Keith Beven and Jim Freer. Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the {GLUE} methodology. *Journal of Hydrology*, 249(1–4):11 – 29, 2001. ISSN 0022-1694. doi: http://dx.doi.org/10.1016/S0022-1694(01)00421-8. URL `http://www.sciencedirect.com/science/article/pii/S0022169401004218`.

FHS Chiew and TA McMahon. Assessing the adequacy of catchment streamflow yield estimates. *Soil Research*, 31(5):665–680, 1993.

P Krause, DP Boyle, and F Bäse. Comparison of different efficiency criteria for hydrological model assessment. *Advances in Geosciences*, 5(5):89–97, 2005.

JEa Nash and JV Sutcliffe. River flow forecasting through conceptual models part ia discussion of principles. *Journal of hydrology*, 10(3):282–290, 1970.

Murray C Peel, Francis H.S Chiew, Andrew W Western, and Thomas A McMahon. Extension of unimpaired monthly streamflow data and regionalisation of parameter values to estimate streamflow in ungauged catchments. Technical report, Centre for Environmental Applied Hydrology, The University of Melbourne, 2000.

A Modified SIMHYD model

```


##### SIMHYD in R
##### MJB, 2014

##### INPUT DATA
# Daily rainfall
# Daily potential evapotranspiration
# Model parameters include:

# insc - Interception store capacity (mm)
# coeff - Maximum infiltration loss (mm)
# sq - Infiltration loss exponent
# smsc - Soil moisture store capacity (mm)
# sub - Constant of proportionality in interflow equation
# crak - Constant of proportionality in groundwater recharge equation
# k - Baseflow linear recession parameter
# psize - Pond area as a fraction of catchment area.
#c_area  - catchment area (km2). This is to work out flows into the pond

##### DEVELOPMENT NOTES
# Model was developed based on the diagram from: "/Users/mjburns/Dropbox/UNGAUGED/useful_docs/57449_1.pdf"
# Value ranges from "hessd-8-8701-2011-print.pdf"

##### DEFINE THE FUNCTION
simhyd <- function(rainfall_input, et_input, insc, coeff, sq, smsc, sub, crak, k, psize, c_area) {

# The input to SIMHYD includes daily rainfall, daily et, and the model parameters
# Give the user warnings if any paramaters are outside safe ranges
if(insc < 0.5 | insc > 5) cat("insc", "is outside the range of typical values", "\n")
if(coeff < 50 | coeff > 400) cat("coeff", "is outside the range of typical values", "\n")
if(sq < 0 | sq > 6) cat("sq", "is outside the range of typical values", "\n")
if(smsc < 50 | smsc > 500) cat("smsc", "is outside the range of typical values", "\n")
if(sub < 0 | sub > 1) cat("sub", "is outside the range of typical values", "\n")
if(crak < 0 | crak > 1) cat("crak", "is outside the range of typical values", "\n")
if(k < 0.003 | k > 0.3) cat("k", "is outside the range of typical values", "\n")
if(psize >= 0) cat("psize", "is a new parameter and no known range exists", "\n")

#Predefine the size of vector, this is based on the input data
size_vector <- NROW(rainfall_input)

#Allocate stores
soil_moisture_store <- rep(NA, size_vector)   #Soil moisture store
ground_store <- rep(NA, size_vector)     #Groundwater store
pond_store <- rep(NA, size_vector) #The pond store
total_flow <- rep(NA, size_vector) #Runoff store
pond_overflow <- rep(NA, size_vector) #Pond overflow vector
final_flow <- rep(NA, size_vector) #Final flow

#Pond details
#The maximum depth and total volume
#As psize is the pond area as a fraction of catchment area
#The volume is depth * psize * catchment area in m2

pond_depth <- 500 #Depth of pond, hardcoded in units of mm
```

```
pond_volume_cubic_m <- (pond_depth/1000) * psize *  (c_area * 10^6)

# Start the rainfall-runoff model
for(i in 1:size_vector) {
  # Need to deal with the initial time-step

    if(i == 1) {
    #Make the soil moisture store initial value half of capacity (value given)
    #this is similar to the chiew spreadsheet
    soil_initial <- 0.5 * smsc

    #Ground initial (in looking at Chiew spreadsheet, this is the value they use)
    ground_initial <- 5

    #Pond storage initial (make it 50% of the maximum available)
    pond_initial <- pond_volume_cubic_m * 0.5

    #Calculate imax (maximum interception)
  imax <- min(insc, et_input[i])

  #Calculate int (interception)
  int <- min(imax, rainfall_input[i])

  #Calculate inr (rainfall not lost in the interception store)
  inr <- rainfall_input[i] - int

  #Calculate rmo (water infiltrated)
  rmo <- min((coeff*exp(-sq*(soil_initial/smsc))), inr)

  #Calculate irun (infiltration excess runoff)
  irun <- inr - rmo

  #Calculate srun (saturation excess runoff and interflow)
  srun <- sub * (soil_initial/smsc) * rmo

  #Calculate rec (water going to the groundwater store)
  rec <- crak * (soil_initial/smsc) * (rmo-srun)

  #Calculate water going into moisture store
  smf <- rmo - srun - rec

  #Add water going to moisture store (smf) to the store
  #Need to account for overflow
  soil_moisture_store[i] <- ifelse(((soil_initial + smf) > smsc), smsc, soil_initial + smf)
  any_overflow <- ifelse(((soil_initial + smf) > smsc), ((soil_initial + smf) - smsc), 0)

  #Calculate pot (account for water lost due to interception)
  pot <- et_input[i] - int

  #Calculate et (assume that et is taken out of the store after any overflow)
  et <- min((10*(soil_initial/smsc)), pot)

  #Reduce the soil moisture store by ET
  soil_moisture_store[i] <- soil_moisture_store[i] - et

  #Add recharge and any overflow from the soil moisture store to the groundwater store
  ground_store[i] <- ground_initial + rec + any_overflow

  #Calculate baseflow
  base <- k * ground_store[i]

  #Take baseflow out from store
  ground_store[i] <- ground_store[i] - base
```

```

#Add all the flow compon together
total_flow[i] <- base + srun + irun

#Calculate flow into pond, in units m3
#The unit conversions: turn flow from mm into m and then multiply by catchment area (km2 * 10^6 m2/km2)
flow_into_pond <- (total_flow[i]/1000) * (c_area*10^6)

#Now try using the pond
pond_store[i] <- ifelse(((pond_initial + flow_into_pond) > pond_volume_cubic_m), pond_volume_cubic_m, pond_initial + flow_into_pond)
pond_overflow[i] <- ifelse(((pond_initial + flow_into_pond) > pond_volume_cubic_m), ((pond_initial + flow_into_pond) - pond_volume_cubic_m), 0)

#Take out some pond water based on pet?
#Work out how much pet can be taken out of the pond
#The calculation is et in mm convert to m, then * pond size (m2)
pond_et_potential <- (et_input[i]/1000) * (psize * (c_area*10^6))

#Take out the ET
pond_store[i] <- pond_store[i] - min(pond_et_potential, pond_store[i])

#The flow to report is the pond overflow in units of mm/day
#need to convert from m3/day into mm/day
#Divided the pond overflow by catchment area in m2, then * 1000 to get mm
final_flow[i] <- (pond_overflow[i]/(c_area*10^6))*1000

  #End the initial timestep

  }

else {

#Calculate imax (maximum interception)
imax <- min(insc, et_input[i])

#Calculate int (interception)
int <- min(imax, rainfall_input[i])

#Calculate inr (rainfall not lost in the interception store)
inr <- rainfall_input[i] - int

#Calculate rmo (water infiltrated)
rmo <- min((coeff*exp(-sq*(soil_moisture_store[i-1]/smsc))), inr)

#Calculate irun (infiltration excess runoff)
irun <- inr - rmo

#Calculate srun (saturation excess runoff and interflow)
srun <- sub * (soil_moisture_store[i-1]/smsc) * rmo

#Calculate rec (water going to the groundwater store)
rec <- crak * (soil_moisture_store[i-1]/smsc) * (rmo-srun)

#Calculate water going into moisture store
smf <- rmo - srun - rec

#Add water going to moisture store (smf) to the store
#Need to account for overflow
soil_moisture_store[i] <- ifelse(((soil_moisture_store[i-1] + smf) > smsc), smsc, soil_moisture_store[i-1] + smf)
any_overflow <- ifelse(((soil_moisture_store[i-1] + smf) > smsc), ((soil_moisture_store[i-1] + smf) - smsc), 0)

#Calculate pot (account for water lost due to interception)
pot <- et_input[i] - int

```

```r
#Calculate et (assume that et is taken out of the store after any overflow)
et <- min((10*(soil_moisture_store[i]/smsc)), pot)

#Reduce the soil moisture store by ET
soil_moisture_store[i] <- soil_moisture_store[i] - et

#Add recharge and any overflow from the soil moisture store to the groundwater store
ground_store[i] <- ground_store[i-1] + rec + any_overflow

#Calculate baseflow
base <- k * ground_store[i]

#Take baseflow out from store
ground_store[i] <- ground_store[i] - base

#Add all the flow compon together
total_flow[i] <- base + srun + irun

#Calculate flow into pond, in units m3
#The unit conversions: turn flow from mm into m and then multiply by catchment area (km2 * 10^6 m2/km2)
flow_into_pond <- (total_flow[i]/1000) * (c_area*10^6)

#Now try using the pond
pond_store[i] <- ifelse(((pond_store[i-1] + flow_into_pond) > pond_volume_cubic_m), pond_volume_cubic_m, pond_store[i-1] + flow_into_pond)
pond_overflow[i] <- ifelse(((pond_store[i-1] + flow_into_pond) > pond_volume_cubic_m), ((pond_store[i-1] + flow_into_pond) - pond_volume_cubic_m), 0)

#Take out some pond water based on pet?
#Work out how much pet can be taken out of the pond
#The calculation is et in mm convert to m, then * pond size (m2)
pond_et_potential <- (et_input[i]/1000) * (psize * (c_area*10^6))

#Take out the ET
pond_store[i] <- pond_store[i] - min(pond_et_potential, pond_store[i])

#The flow to report is the pond overflow in units of mm/day
#need to convert from m3/day into mm/day
#Divided the pond overflow by catchment area in m2, then * 1000 to get mm
final_flow[i] <- (pond_overflow[i]/(c_area*10^6))*1000

# End the non first time-step

}
# now end total loop
}

#End the function now and return total flow
return(final_flow) }
```

SIMHYD_IN_R_5_revised.R