Themen

Programm zur Anwendung von TUW

```
## Load the data
data(example_TUWmodel)

## Simulate runoff and plot observed vs simulated series
## Lumped case (weighted means of the inputs)
simLump <- TUWmodel(prec=apply(P_Vils, 1, weighted.mean, w=areas_Vils),
                    airt=apply(T_Vils, 1, weighted.mean, w=areas_Vils),
                    ep=apply(PET_Vils, 1, weighted.mean, w=areas_Vils),
                    area=sum(areas_Vils),
            param=c(1.02,1.70,2,0,-0.336,
                    0.934,121,2.52,
                    0.473,9.06,142,
                    50.1,2.38,10,25))

plot(as.Date(names(Q_Vils)), Q_Vils, type="l", xlab="", ylab="Discharges
[mm/day]")
 lines(as.Date(rownames(T_Vils)), simLump$q, col=2)
legend("topleft", legend=c("Observations","Simulations"), col=c(1,2), lty=1,
bty="n")

plot(as.Date(rownames(SWE_Vils)), apply(SWE_Vils, 1, weighted.mean,
w=areas_Vils),
     type="l", xlab="", ylab="Snow Water Equivalent [mm]")
 lines(as.Date(rownames(T_Vils)), simLump$swe, col=2)

## Distribute input case (6 zones)
simDist <- TUWmodel(prec=P_Vils, airt=T_Vils, ep=PET_Vils,
area=areas_Vils/sum(areas_Vils),
            param=c(1.02,1.70,2,0,-0.336,
                    0.934,121,2.52,
                    0.473,9.06,142,
                    50.1,2.38,10,25))

plot(as.Date(names(Q_Vils)), Q_Vils, type="l", xlab="", ylab="Discharges
[mm/day]")
 lines(as.Date(rownames(T_Vils)), simDist$q, col=2)
legend("topleft", legend=c("Observations","Simulations"), col=c(1,2), lty=1,
bty="n")

plot(as.Date(rownames(SWE_Vils)), apply(SWE_Vils, 1, weighted.mean,
w=areas_Vils),
     type="l", xlab="", ylab="Snow Water Equivalent [mm]")
 lines(as.Date(rownames(T_Vils)), apply(simDist$swe, 1, weighted.mean,
w=areas_Vils), col=2)
```

```
## Distributed input and parameters case
parametri <- matrix(rep(c(1.02,1.70,2,0,-0.336,
                          0.934,121,2.52,
                          0.473,9.06,142,
                          50.1,2.38,10,25), 6), ncol=6)
parametri[2,] <- c(1.4, 1.7, 1.9, 2.2, 2.4, 3.0)
simDist2 <- TUWmodel(prec=P_Vils,
                     airt=T_Vils,
                     ep=PET_Vils,
                     area=areas_Vils/sum(areas_Vils),
             param=parametri)

plot(as.Date(names(Q_Vils)), Q_Vils, type="l", xlab="", ylab="Discharges
[mm/day]")
 lines(as.Date(rownames(T_Vils)), simDist2$q, col=2)
legend("topleft", legend=c("Observations","Simulations"), col=c(1,2), lty=1,
bty="n")
```

Programm für numerische Berechnung eines Einzellinearspeichers,

```
```{r ELSnum}

Nt <- replicate(30,0)
Nt[7] <- 25
Nt[18]<- 43

# Parameter des ELSnum
k <- 0.35
a <- 1/k # Einheiten, dies es braucht, bis der Speicher um eine log Einheit
fällt
S0<- 0

ELSnum <- function(a=10,S0=0,Nt){
  LN <- length(Nt)
  St <- numeric(LN)
  Qt <- numeric(LN)

  i=1
  while(i<=LN){
    if(i==1){
      St[1] <- S0+Nt[1]
      Qt[1] <- 0
    }else{
      Qt[i] <- St[i-1]*1/a
      St[i] <- St[i-1]-Qt[i]+Nt[i]
    }
    i <- i+1
  }
  Resultat <- data.frame(Nt,St,Qt)
  return(Resultat)
}
```
```

```
ELSnum(5,0,Nt)
```

```{r ELS, echo=FALSE, fig.cap="ELS-Modell"}
ELS <- function(N=50,C=0.1,K=7,ta=0,te=10,tm=30,dt=1){
  errorflag <- FALSE

  # Prüfe auf Fehler
  if (N<0.0){
    error <- "N ist < 0.0"
    errorflag <- TRUE
  }
  if(C < 0.0 | C > 1.0){
    error <- "C nicht zwischen 0.0 und 1.0"
    errorflag <- TRUE
  }
  if(K <= 0.0){
    error <- "K nicht >= 0"
    errorflag <- TRUE
  }
  if (ta<0.0){
    error <- "ta ist < 0.0"
    errorflag <- TRUE
  }
  if (ta>=te){
    error <- "ta ist > te"
    errorflag <- TRUE
  }
  if (ta>=tm){
    error <- "ta ist > tm"
    errorflag <- TRUE
  }
  if (dt<=0.0){
    error <- "dt ist <= 0.0"
    errorflag <- TRUE
  }
  if (dt>te){
    error <- "dt ist > te"
    errorflag <- TRUE
  }

  # Erzeuge die Zeitreihe
  tv <- seq(ta,tm,by=dt) # Zeitvariable in Tagen
  # Berechne die Länge in Tagen
  Lt <- length(tv)
  # Initialisiere die Abflussserie mit 0, Einheit mm
  Qt <- replicate(Lt,0)
  # Erzeuge Variable für letzten Wert der Schleife
  QL <- 0.0 # last discharge
```

```
  # i ist die Integer Laufvariable, t ist die Zeit
  i  <- 1
  t  <- ta
  while (i<=Lt) {
    if(t<=te){
      # Anstieg (t < te)
      Qt[i] <- (N*C)*(1-exp(-t/K))
      QL   <- Qt[i]
      } else {
      # Rückgang
      Qt[i] <- QL*exp(-(t-te)/K)
    }
    # Gehe einen Zeitschritt dt weiter für t und setze Index i um +1 hoch
    t <- t + dt
    i <- i + 1
  }

  # plot
  if (errorflag){
    print(error)
  } else{
    plot(tv,Qt, col=2, xlab="Zeit (Tage)", ylab="Q (l/s)", type = "o")
  }
}
# Ausführen der Funktion ELS mit Standardparametern
ELS()
```