Natural Resources Management

Streamflow forecast model for Ebro river and dam impact analysis

Students:

Luca Acquati 920545

Mostafa Baghdadi 952057

Thomas Manzoni 920635

Ghodrat Rezaei 952128

Part 2

Numerical comparison of the performance attained by few project alternatives on different stakeholders' interests (flooding and irrigation).

Assuming a linear function for natural lake release, we finally dedicate by trial and error a coefficient of 20000 for linear function (r = 20000 h). This linear function is just a simplification of the max release of the dam (when the dam is completely open it would behave like a natural lake). Generally the discrete operation zone is limited by two physical and normative constraints:

1. **Physical constraints:** setting the value of minimum and maximum release (as what has been mentioned): min r = 0 (no water storage leads to no release), max = 20000 h.

2**. Normative constraints:** setting the value of h_min value equal to 0 and h_max value equal to the designed level of the dam.

It has also been assumed for the water demand a flow of 65 [m3/s] and for flooding threshold a level of 0.04 [m], leading to the following and final conclusions. To be noticed that that all above assumptions are based on trial and error and logical guess.

Area of lake is given, which is 25729870000 [m2].

PART 2.1: performance of Alternative-0 (i.e. no dam)

Regulating_release function

```
function r = regulating_release( param , h )
```

```
% LAKE MODELS PARAMETERS

% natural storage-discharge relationship

alfa     = param.nat.alpha   ;
h0       = param.nat.h0     ; % [m]

% regulated storage-discharge relationship
h_min    = param.reg.h_min  ; % [m]
h1       = param.reg.h1     ; % [m]
h2       = param.reg.h2     ; % [m]
h_max    = param.reg.h_max  ; % [m]
m1       = param.reg.m1     ; % [m2/s]
m2       = param.reg.m2     ; % [m2/s]
w        = param.reg.w      ; % [m3/s]
h3       = param.reg.h3     ; % [m]

% COMPUTATION OF THE LAKE RELEASE

% 1) regulated storage-discharge relationship
% water saving
L1 = w + m1 * ( h - h1 ) ;
% floods control
L2 = w + m2 * ( h - h2 ) ;
% release
r  = max( [ min( L1 , w ) ; L2 ] )  ;

% 2) normative constrains
r( h <= h_min ) = 0 ;                                  ;    %
completely closed dam gates
r( h >= h_max ) = alfa * h( h >= h_max );    % completely open dam
gates
r( h <= h3 ) = 0 ;
% 3) physical constrains --> the values of the parameters h1, h2, m1,
m2, w might generate
% release values that are not admissible from a physical point of view

% the release can not be negative
r( r < 0 )  = 0                             ;
% find the release values larger than the maximum admissible outflow
idx         = r > alfa * h     ;
r( idx )    = alfa* h(idx) ;
% the release must be 0 if h(t) < h0 (this constrain is necessary if
% 'h_min' is lower than 'h0')
r( h < h0 ) = 0                             ;
```

Simulating_lake function

```
function [s, h, r] = simulating_lake( n, h_init, param )
delta = 60*60*24;
H = length(n)-1 ; % 10 years horizon
% initialization of vectors
s = nan( size(n) );
h = nan( size(n) );
r = nan( size(n) );
% initial condition t=1
h(1) = h_init;
s(1) = h_init * param.nat.S ;
for t=1:H
   % 1)release
   r(t+1) = regulating_release( param , h(t) );
   %r(t+1) = param.nat.beta*( h(t) - param.nat.h0 ).^param.nat.alpha;
% this replaces the previous line in case of simulation of the natural
lake
   % 2) mass-balance
   s(t+1) = s(t) + ( n(t+1) - r(t+1) )*delta ;
   % 3) s->h
   h(t+1) = s(t+1)/param.nat.S ;
end
```

Simulating_nat_lake

```
function [s, h, r] = simulate_nat_lake( n, h_init, param )
delta = 60*60*24;
H = length(n)-1 ; % 10 years horizon
% initialization of vectors
alpha = param.nat.alpha;
s = nan( size(n) );
h = nan( size(n) );
r = nan( size(n) );
% initial condition t=1
```

```
h(1) = h_init;
s(1) = h_init * param.nat.S ;
for t=1:H
   % 1)release
   %r(t+1) = regulated_release( param , h(t) );
   r(t+1) = alpha * h(t);  % this replaces the previous line in case
of simulation of the natural lake
   % 2) mass-balance
   s(t+1) = s(t) + ( n(t+1) - r(t+1) )*delta ;
   % 3) s->h
   h(t+1) = s(t+1)/param.nat.S ;
end
```

Simulating_reg_lake

```
function [s, h, r] = simulating_reg_lake( n, h_init, param )
delta = 60*60*24;
H = length(n)-1 ; % 10 years horizon
% initialization of vectors
s = nan( size(n) );
h = nan( size(n) );
r = nan( size(n) );
% initial condition t=1
h(1) = h_init;
s(1) = h_init * param.nat.S ;
for t=1:H
   % 1)release
   r(t+1) = regulating_release( param , h(t) );
   %r(t+1) = param.nat.beta*( h(t) - param.nat.h0 ).^param.nat.alpha;
% this replaces the previous line in case of simulation of the natural
lake
   % 2) mass-balance
   s(t+1) = s(t) + ( n(t+1) - r(t+1) )*delta ;
   % 3) s->h
   h(t+1) = s(t+1)/param.nat.S ;
end
```

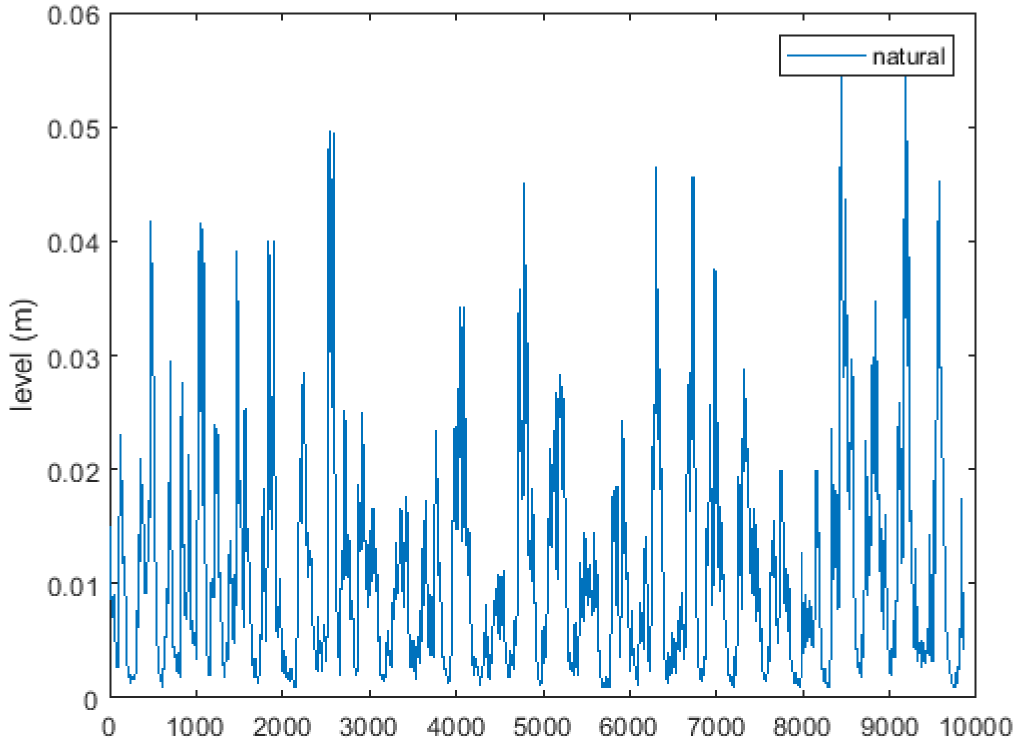
```
load -ascii Tudela.csv
q = Tudela(:,5); % average daily stream flow[m3/s]
h_init = 0.015 ; % initial condition
```

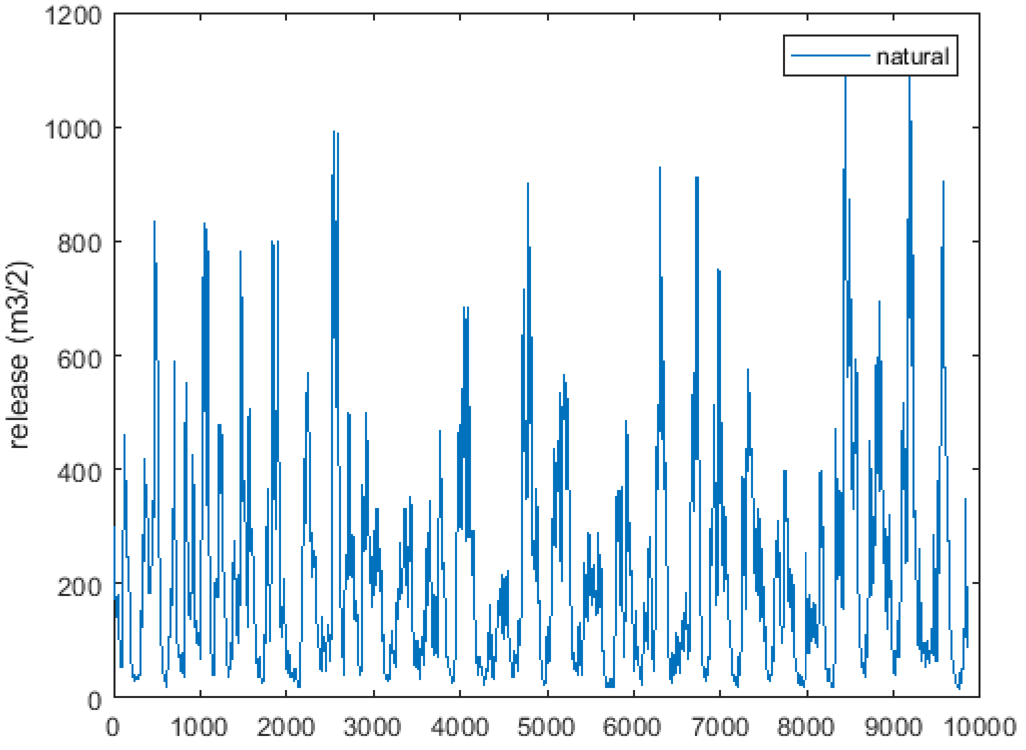
```
u = Tudela(:,4); %average daily precipitation[mm/d]
t = Tudela(:,6); %average daily temperature[c]
w = 65; %  assume 65 m3/s as water demand for irrigation
h_flo = 0.04; % assume 0.04 m as flood threshold

% natural level-discharge relationship:
param.nat.S = 25729870000 ; % [m2]
param.nat.alpha = 20000 ; %[m2/s]
param.nat.h0 = 0 ;   %[m]


%reference of natural lake (since Lake Tudela  was a natural system)
n = q;
n = [ nan; n ] ; % for time convention
[s_nat, h_nat, r_nat] = simulating_nat_lake( n, h_init, param ) ;

figure;
plot(h_nat); ylabel('level (m)'); legend('natural')
figure;
plot(r_nat); ylabel('release (m3/2)'); legend('natural')

h_nat = h_nat(2:end);
r_nat = r_nat(2:end);

% I1: daily average squared deficit
dn = max( w-r_nat, 0 ) ;
I1_nat = mean( dn.^2 )      %  Since our level value is between 0 and
1,
                            %  the area indicator becomes negative
value,
                            % therefore we dont use area indicator
                            % {S _flo( idx ) = 0.081*h_reg(idx).^3
                            % -0.483*h_reg(idx).^2 +1.506*h_reg(idx)-
1.578}




% IF1 = mean flooded surface in the city of Locarno
Ny = length(h_nat)/365;
I2_nat = sum( h_nat>h_flo )/Ny
```

```
I1_nat =   144.2188

I2_nat =  5.2593
```

＊＊＊Since level value is between 0 and 1, the area indicator becomes negative value, therefore are indicator has not been used:

```
{S _flo( idx ) = 0.081*h_reg(idx).^3-0.483*h_reg(idx).^2 +1.506*h_reg(idx)-1.578}
```


0.06
0.05
0.04
0.03
0.02
0.01
0
level (m)
natural
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


1200
1000
800
600
400
200
0
release (m3/2)
natural
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

PART 2.2: Active capacity of the dam

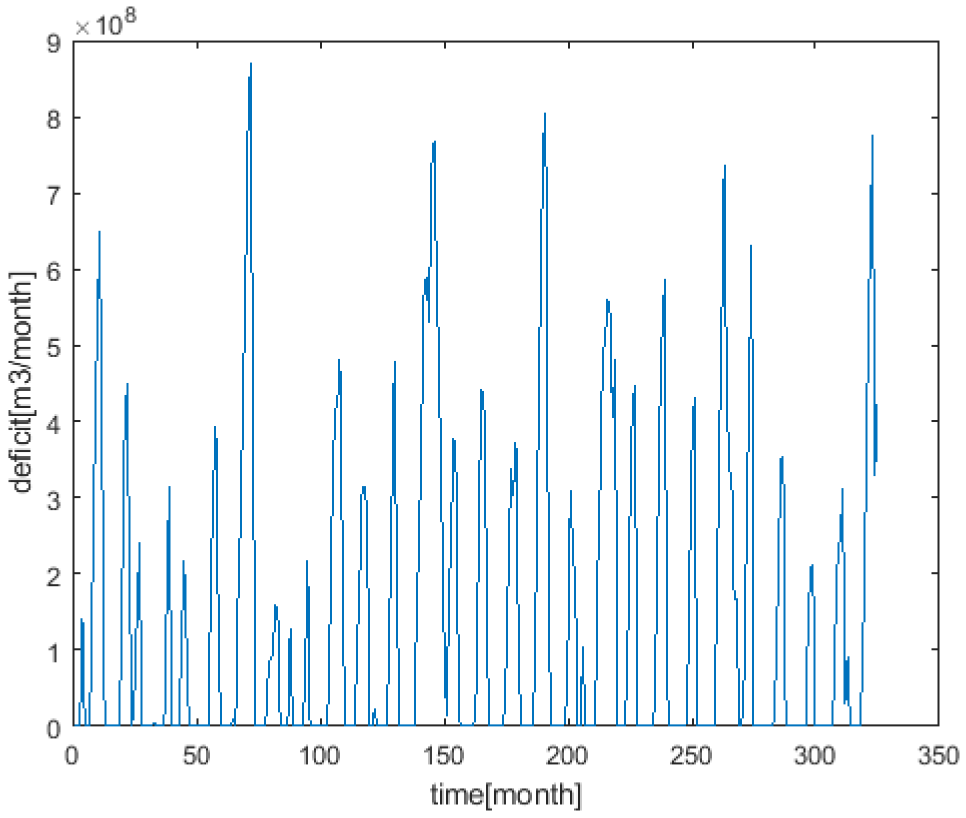
Daily to monthly function

```
function xMonth = dailyToMonthly( x, N )

% function qMonth = dailyToMonthly( x, N )
%
% function to convert a daily timeseries into monthly average values
% input:    - x = trajectory of daily values (vector of 365*N elements)
%           - N = number of years
%
% output:   xMonth = monthly average values (matrix 12*N)

% reshape vector into matrix
xx = reshape(x,365,N);
% id of months
dm = [1*ones(31,1); 2*ones(28,1); 3*ones(31,1); 4*ones(30,1); 5*ones(31,1);
6*ones(30,1); 7*ones(31,1);...
    8*ones(31,1); 9*ones(30,1); 10*ones(31,1); 11*ones(30,1); 12*ones(31,1)] ;
% monthly average
xMonth = nan(12,N);
for i = 1:12
    for j = 1:N
        idx = dm == i;
        x_ = xx(idx,j) ;
        xMonth(i,j) = mean( x_ ) ;
        clear x_
    end
end

end
```

```
% monthly mean flow
T = 27;  %number of years
qMonth = dailyToMonthly(q, T); % m3/s

% target release = downstream water demand w
w = 100 ; % m3/s
```

```
% Sequent Peak Analysis
deltaT = 3600*24*[31 28 31 30 31 30 31 31 30 31 30 31]';
Q = qMonth(:).*repmat(deltaT,27,1) ; % m3/month
W = w*ones(size(Q)).*repmat(deltaT,27,1) ; % m3/month
K = zeros(size(Q));
K(1) = 0;

for t = 1:length(Q)
    K(t+1) = K(t) + W(t) - Q(t) ;
    if K(t+1) < 0
        K(t+1) = 0 ;
    end
end

figure; plot( K );xlabel('time[month]'); ylabel('deficit[m3/month]');
Kopt = max(K);
% once the capacity is defined, it's necessary to set dam
height/surface
% In the case of Lake Tudela, the surface is 25729.870000 km2
% so we can make the following check:
h_max = Kopt/25729870000
% this dam height value is in the range of different level(comparison
to the Lab case_studies) because of it's big area,
% suggesting that the storage capacity of Lake Tudela, although
designed
% by nature and not artificially, is suitable for providing water
supply
% to the downstream users
```

```
h_max =  0.0338  [m]
```

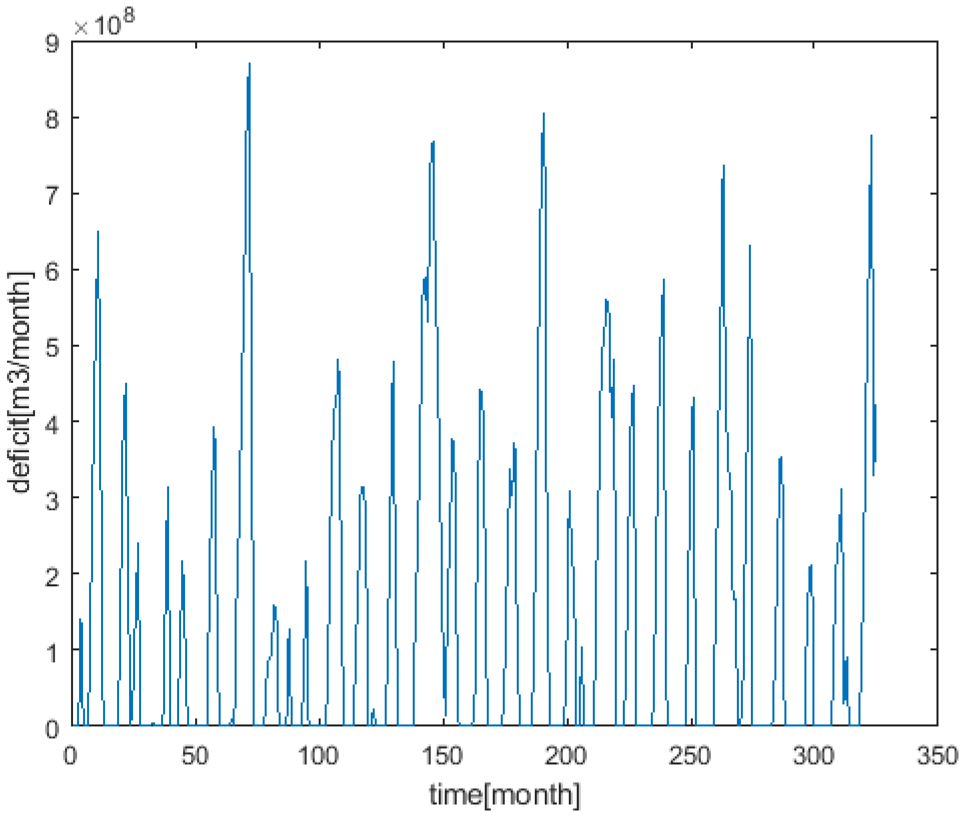


PART 2.3.a: simulation of water reservoir under regulation

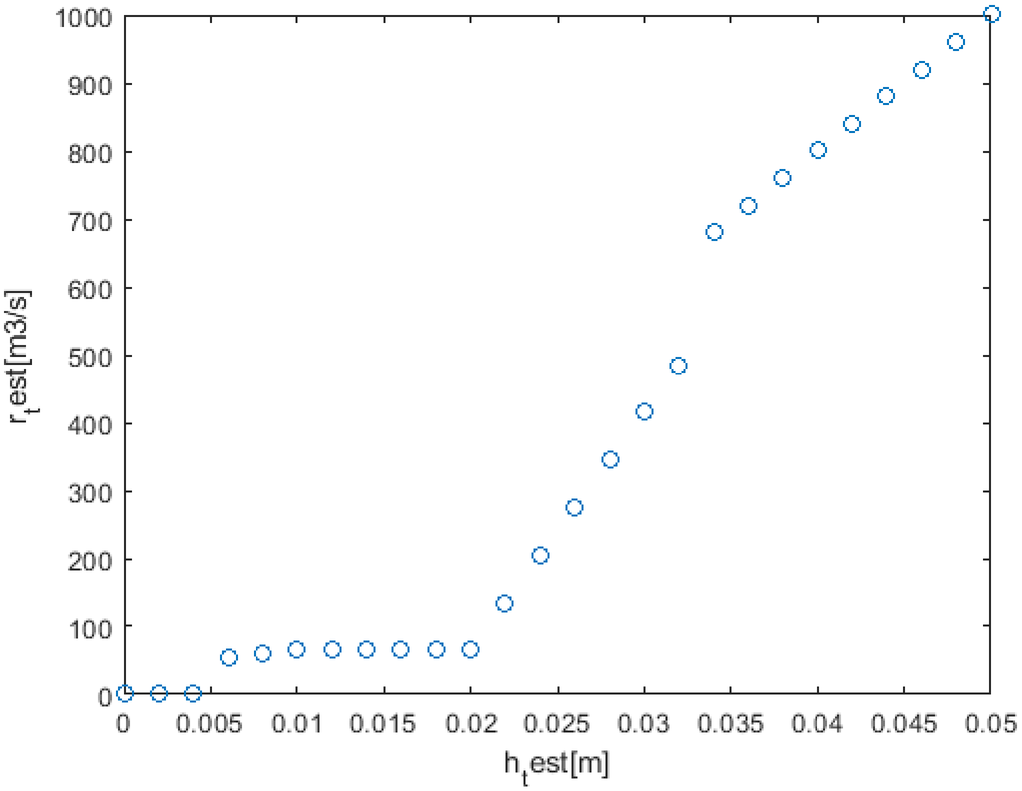
```
% regulated level-discharge relationship:
param.reg.w = 65 ;    %[m3/s]
param.reg.h_min = 0 ; %[m]
param.reg.h_max = h_max ; %h_max is the height of active dam [m]
param.reg.h1 = 0.01 ; %[m]
param.reg.h2 = 0.02 ; %[m]
param.reg.m1 = 3000 ;  %[m2/s]
param.reg.m2 = 35000 ; %[m2/s]
param.reg.h3 = 0.005 ; %[m]


% test regulated level-discharge relationship
h_test = [ 0:0.002:0.05 ] ;
r_test = regulating_release( param , h_test );
figure; plot(h_test,r_test, 'o');
xlabel('h_test[m]'); ylabel('r_test[m3/s]');
```

```
% simulation of lake dynamics
n = q;
n = [ nan; n ] ; % for time convention
h_init = 0.015 ; % initial condition


[s, h, r] = simulating_lake( n, h_init, param ) ;

figure; plot( h ); xlabel('time[d]'); ylabel('water level[m]');

figure; plot(r) ; xlabel('time[d]'); ylabel('release[m3/s]');

% impacts
h1 = h(2:end);
r1 = r(2:end);
w = param.reg.w ;
h_flo = 0.04 ;

% daily average squared deficit
def = max( w-r1, 0 );
Jirr = mean( def.^2 )

% daily average flooded area
Ny = length(h1)/365;
jflo = sum( h>h_flo )/Ny
```

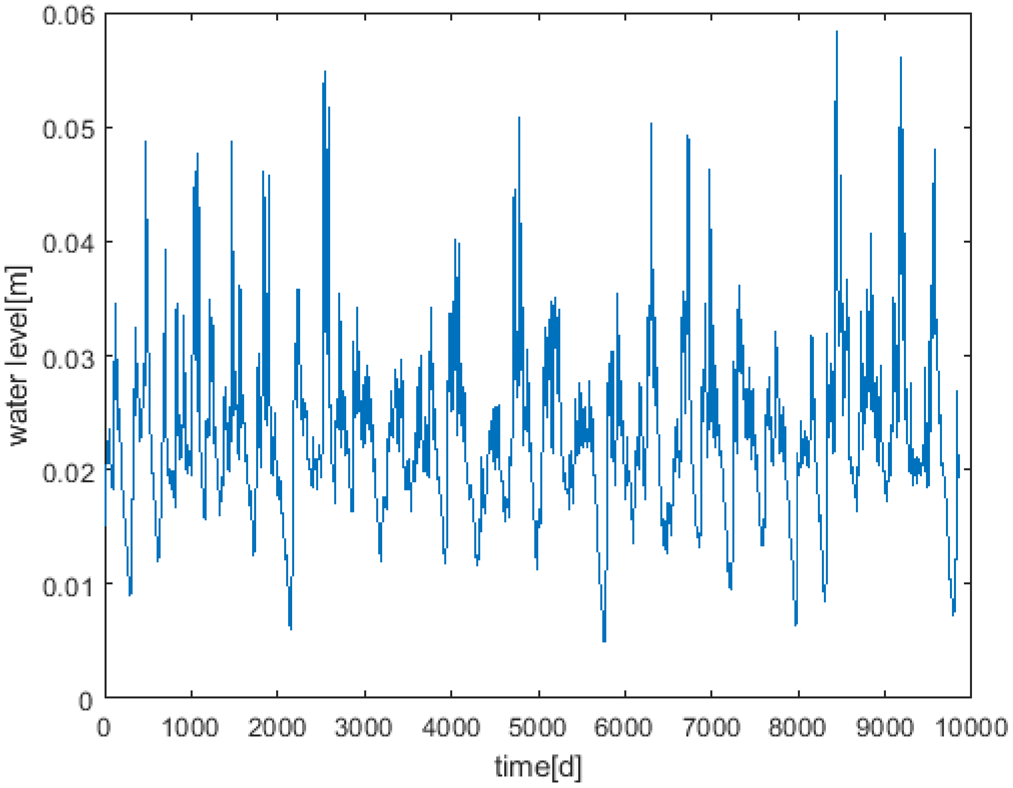
```
Jirr =

    2.3771


jflo =

   10.4074
```

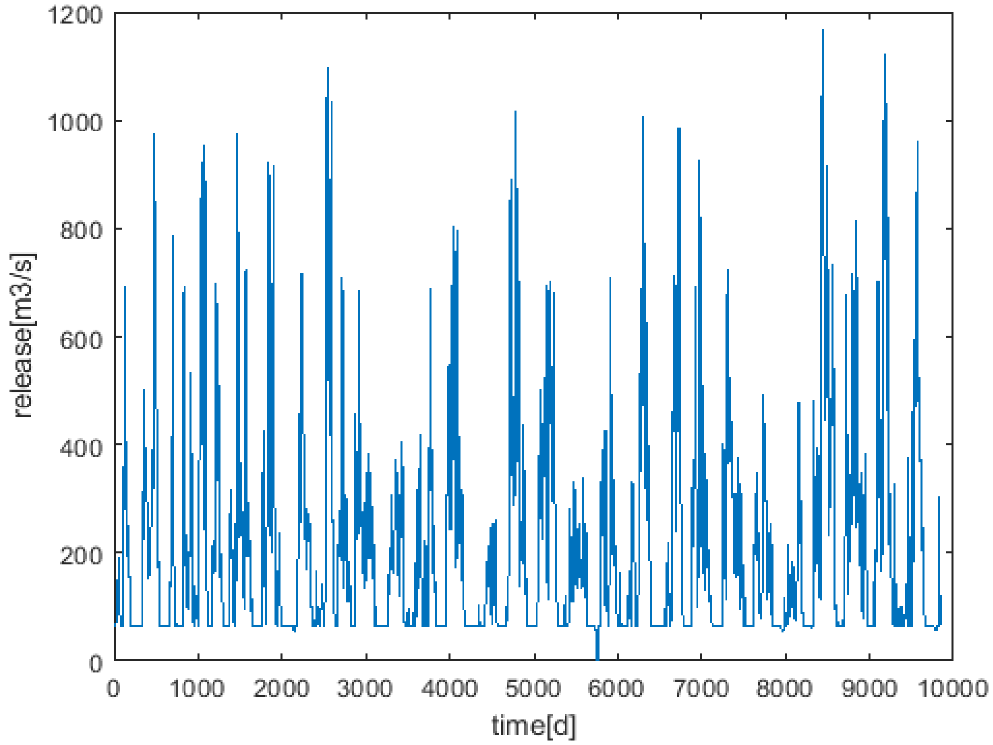


×10^8
deficit[m3/month]
time[month]
0
50
100
150
200
250
300
350
0
1
2
3
4
5
6
7
8
9



r_test[m3/s]
h_test[m]
0
100
200
300
400
500
600
700
800
900
1000
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05


0.06
0.05
0.04
0.03
0.02
0.01
0
water level[m]
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
time[d]


1200
1000
800
600
400
200
0
release[m3/s]
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
time[d]

PART 2.3.b: Performance of few management alternatives defined by different parameterizations of the Standard Operating Policy

Evaluating_objective function

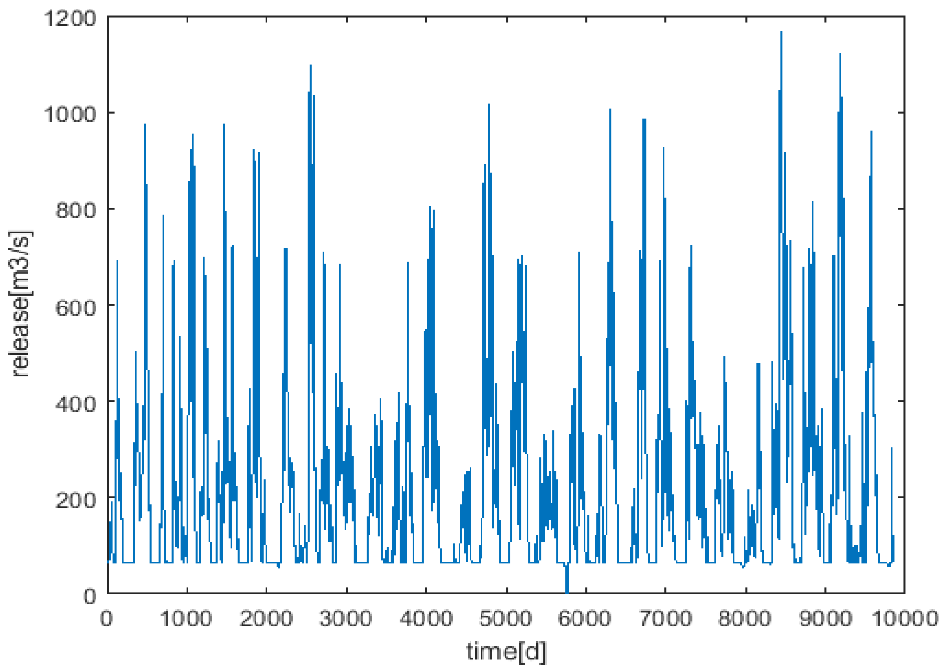
```
function [ Jirr, Jflo ] = evaluating_objective(x, M, V)
%
% function f = evaluate_objective(x, M, V)
%
% Function to evaluate the objective functions for the given input
vector x.%
% x is an array of decision variables and f(1), f(2), etc are the
% objective functions. The algorithm always minimizes the objective
% function hence if you would like to maximize the function then
multiply
% the function by negative one. M is the numebr of objective functions
and
% V is the number of decision variables.
%
% This functions is basically written by the user who defines his/her
own
% objective function. Make sure that the M and V matches your initial
user
% input.
%

x = x(1:V) ;
x = x(:)   ;

% --------------------------------------
% insert here your function:
% global variable to pass inside extra inputs
global opt_inputs ;
n = opt_inputs.n ;
h_init = opt_inputs.h_init ;
param = opt_inputs.param;
h_flo = opt_inputs.h_flo;
w = param.reg.w ;
% 1) policy param
param.reg.h1 = x(1) ;
param.reg.h2 = x(2) ;
param.reg.h3 = x(3) ;
param.reg.m1 = x(4) ;
param.reg.m2 = x(5) ;
```

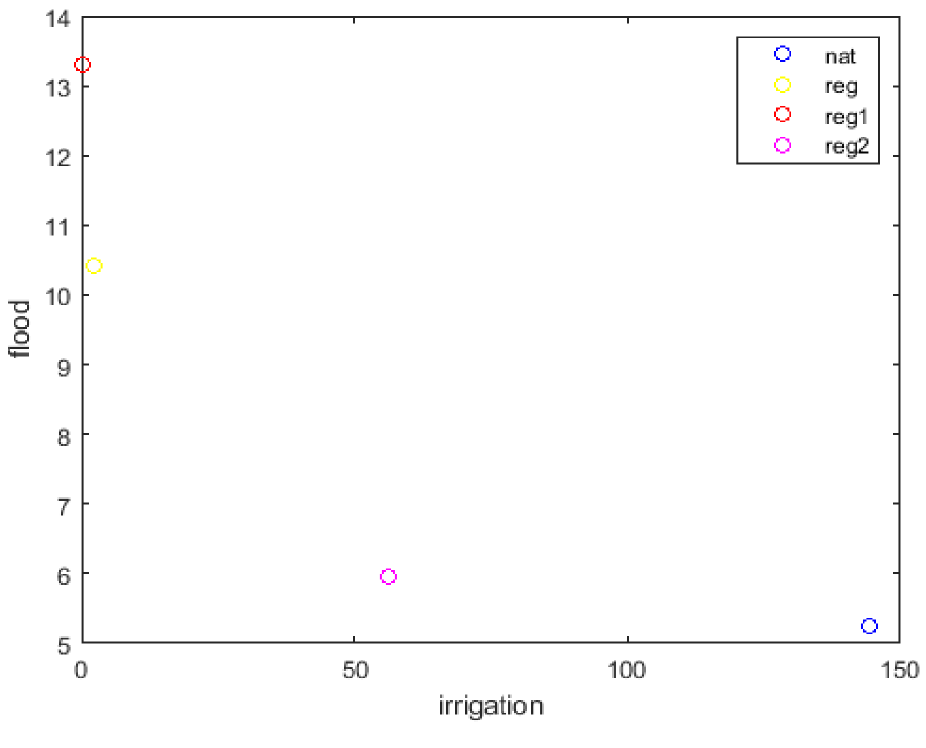
```
% 2) run simulation
[s_reg, h_reg, r_reg] = simulating_reg_lake( n, h_init, param ) ;
% 3) compute 2 objs
h_reg = h_reg(2:end);
r_reg = r_reg(2:end);
% I1: daily average squared deficit
dr = max( w-r_reg, 0 ) ;
Jirr = mean( dr.^2 ) ;
% IF1 = mean flooded indication in the city of Tudela
Ny = length(h_reg)/365;
Jflo = sum( h_reg>h_flo )/Ny

end
```

```
% evaluating different alternatives in comparison with no-dam
alternatives(natural lake)

global opt_inputs;
opt_inputs.n = n ;
opt_inputs.h_init = h_init ;
opt_inputs.param = param ;
opt_inputs.h_flo = h_flo ;

[ Jirr, Jflo ] = evaluating_objective([0.01 0.02 0.005 3000 35000], 2,
5)        % existing policy indicator
[ Jirr1, Jflo1 ] = evaluating_objective([0.01 0.033 0 3000 35000], 2,
5)         % expected improving water supply indicator
[ Jirr2, Jflo2 ] = evaluating_objective([0.01 0.011 0 5000 50000], 2,
5)    % expected improving flood indicator

figure; plot( I1_nat,I2_nat , 'bo') ;
hold on; plot(  Jirr, Jflo  , 'yo') ;
hold on; plot(  Jirr1, Jflo1 , 'ro') ;
hold on; plot(  Jirr2, Jflo2 , 'mo') ;
xlabel('irrigation'); ylabel('flood');
legend('nat','reg', 'reg1', 'reg2');
```

The alternative-0(no-dam) has the performance of [I1_nat(irrigation) =144.21 and I2_nat(flo) = 5.25]. It has been used manually different parameter value by changing the decision variable (h1 h2 h3 m1 m2) based on different stakeholder's interests(flood, irrigation). The existing policy with decision variable of ([h1=0.01, h2=0.02, h3=0.005, m1=3000, m2=35000]) has the indicator of [Jirr = 2.37 and Jflo = 10.44] , showing that it has high improvement in irrigation and a little worsening of performance of flood. The second policy with decision variable of ([h1=0.01,h2 =0.033, h3=0, m1=3000,m2=35000]) has the indicator of [Jirr1 = 0 and Jflo1 = 13.296] , showing that it has a high improvement in irrigation(which is Jirr1 = 0; this is very interesting!!!!!????) and a considerable worsening of performance of flood(Jflo1 = 13.296) in comparison to alternative_0(Jflo = 5.25). The third policy with decision variable of ([h1=0.01,h2 =0.011, h3=0.005, m1=5000,m2=50000]) has the indicator of [Jirr2 =56.03 and Jflo2 = 5.96], showing that it has very little worsening performance in flood (which is Jflo2 = 5.96)in comparison with alternative_0(J_flo = 5.25) and a high improvement of performance in irrigation(Jirr2 =56.03) in comparison with alternative_0(J_irr = 144.21).

```
Jflo = 10.4074

Jirr = 2.3771



Jflo1 = 13.2963

Jirr1 = 0



Jflo2 = 5.9630

Jirr2 = 56.0290
```


release[m3/s]
0
200
400
600
800
1000
1200
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
time[d]


flood
5
6
7
8
9
10
11
12
13
14
0
50
100
150
irrigation
nat
reg
reg1
reg2