Practical Considerations for the Application of a Recursive Analytical Solution of the Nonlinear Storage Equation for Flood Control Reservoir Sizing

József Szilágyi[1,*]

[1]Department of Hydraulic and Water Resources Engineering, Faculty of Civil Engineering, Budapest University of Technology and Economics, Műegyetem Rakpart 1, Budapest, H-1111 Hungary

[*] Corresponding author e-mail: szilagyi.jozsef@emk.bme.hu

Abstract

In the practice of flood control reservoir sizing the smooth theoretical stage-storage power function relationship necessary for an analytical solution of the nonlinear storage equation is often replaced by a complex series of piece-wise linear functions. A recursive analytical solution can still be employed provided the straight line sections are segmented into a series of intervals over which the straight lines are approximated to a prescribed accuracy by individual power functions. During the application of the recursive analytical solution one must keep track of the outflow stage in order to employ the relevant parameters of the piecewise power functions.

Keywords

nonlinear storage equation, universal recursive analytical solution, discrete linear cascade model, recursion, flood control reservoir

1 Introduction

Recently Szilágyi [1] introduced a recursive analytical solution to the nonlinear storage equation [2]

$$v\kappa[Q(t)]^{v-1}\frac{dQ(t)}{dt}+Q(t)=I(t) \tag{1}$$

in the form [3, 4] of

$$Q(t+dt)=e^{-kdt}Q(t)+(1-e^{-kdt})\left[\frac{1}{kdt}-\frac{e^{-kdt}}{1-e^{-kdt}}\right]I(t)+(1-e^{-kdt})\left[1-\frac{1}{kdt}+\frac{e^{-kdt}}{1-e^{-kdt}}\right]I(t+dt) \tag{2}$$

where $I(t)$ and $Q(t)$ are in- and outflow values of the flood control reservoir at time t. The weir-controlled outlet has a rating curve of

$$Q(t)=c[h(t)]^{n} \tag{3}$$

where h is outflow stage, while storage (S) is related to h as

$$S(t)=a[h(t)]^{m} \tag{4}$$

Here c, n, a, and m are constants, and so are $\upsilon = m / n$ and $\kappa = a / c^{\upsilon}$ in Eq. (1). The time-varying storage coefficient, k, in Eq. (2) is redefined [1] for each time reference, t, incremented by dt as

$$k(t) = \{\upsilon\kappa[Q(t)]^{\upsilon-1}\}^{-1} \ . \qquad (5)$$

In real-life applications when the stage-storage relationship cannot be defined by the smooth power-function of Eq. (4), but rather by a series of piece-wise linear functions, as demonstrated in Fig. 1, a problem arises during the recursive employment of the analytical solution of Eq. (2). Below a practical solution is outlined for such cases.

2 Methods

A hypothetical stage-storage relationship, $S(h)$, is defined in Fig. 1. For clarity, the relationship is made up of only (N =) four line sections. Real-life, practical $S(h)$ curves may contain a significantly higher number of line sections.

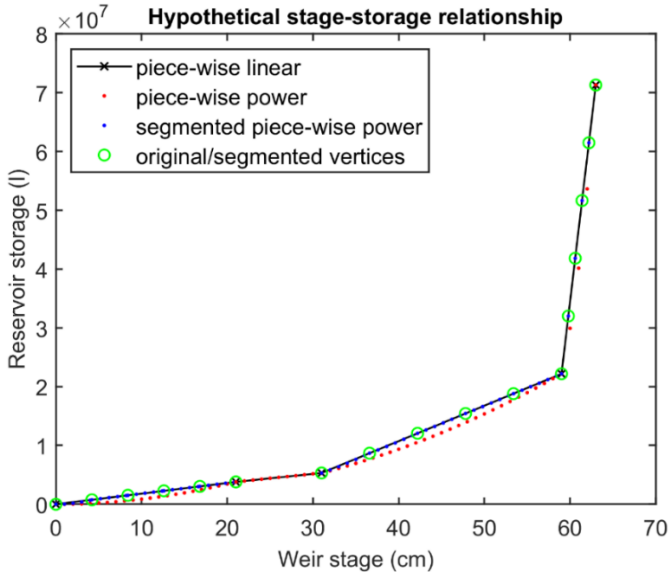

Fig. 1. A hypothetical piece-wise linear $S(h)$ relationship.

Each line section, LS_i (i = 1, … , N), can be approximated by a power function of the form $a_i h^{m_i}$ where the parameters, a_i and m_i, are obtained from evaluating the actual power function at the end points (i.e., vertices) of the corresponding line section. For each line section the evaluation yields

$$m_i = \frac{ln(S_{i+1}) - ln(S_i)}{ln(h_{i+1}) - ln(h_i)} \ , \quad a_i = \frac{S_i}{h_i^{m_i}} \ , \quad i = 1, \ldots, N \qquad (6)$$

where ln denotes the natural logarithm. Note that N line sections require N + 1 vertices [i.e., (h_i, S_i) value pairs] to be specified and that the starting vertex of the stage-storage relationship cannot be zero. In this study an h_1 value of 10^{-6} cm and S_1 of 10^{-8} l were chosen.

Fig. 1 displays the resulting approximations by the red dots. Those line segments where the approximation, upon evaluation over the relevant line section, will not meet a preset limit of maximum allowed difference between the straight line and the power function value, are

further segmented into a predefined number (i.e., five in Fig. 1) of smaller line sections. The number of segmented line sections can be increased (e.g., from five to ten) until the required accuracy is achieved for every line segment. In Fig. 1 this accuracy was arbitrarily set to (S_{max} – S_{min})/100. As seen, the original power-function approximation met the prescribed accuracy only for the second line section (i.e., that without the blue dots). Since the remaining three line sections were each segmented into five smaller sections, in the end M = 16 line sections resulted, altogether yielding M pairs of (a_j, m_j) values.

The recursive application of Eq. (2) can be ensured by keeping track of which line segment (out of M) the actual $h(t)$ belongs to so that the relevant (a_j, m_j) values can be employed in Eq. (5). Note again that M line segments now require M + 1 (h_j, S_j) vertices to be specified for choosing the relevant (a_j, m_j) value pairs.

3 Results and discussion

Application of the 16 different (a_j, m_j) values together with c = 6 l/(s cm$^{1.5}$), n = 1.5, and $Q(0)$ = 0.1 l/s yield high flood peak reduction (Fig. 2) for the specified triangular design flood hydrograph. The outflow stage values span almost the entire weir-stage range displayed in Fig. 1. The sharp slope changes in the stage-storage relationship show up distinctly in the corresponding outflow stage and discharge values by breaking the otherwise smooth curves.

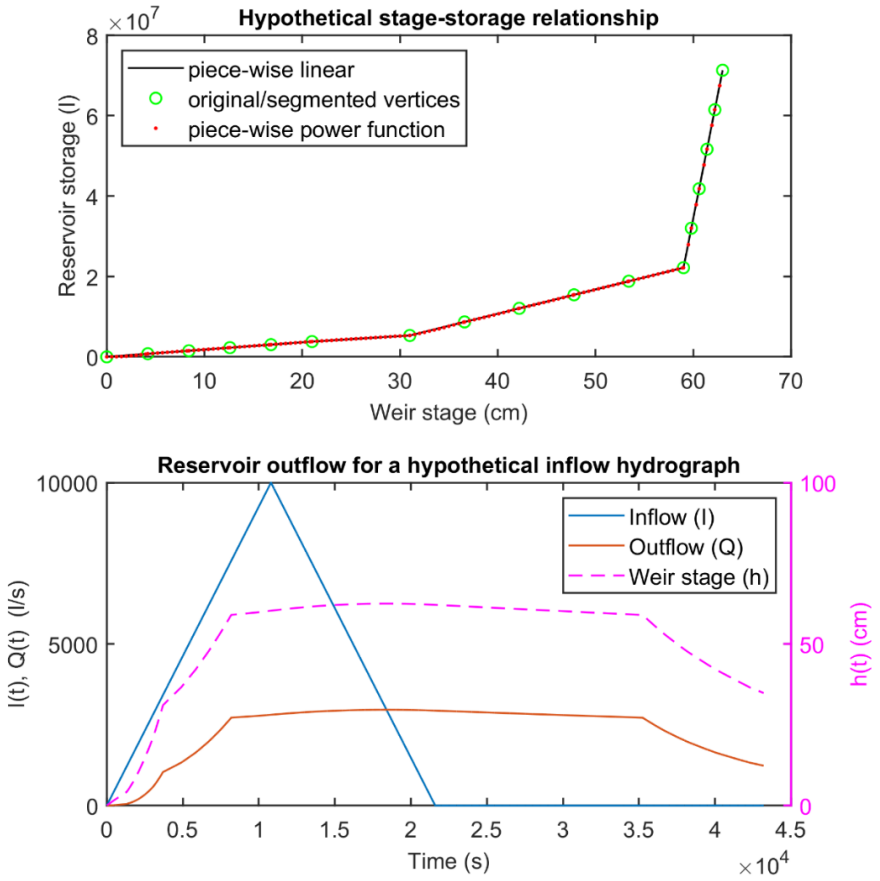

Fig. 2. Time series of design inflow as well as reservoir outflow and weir stage values. Maximum weir stage is 62.54 cm.

When the convex $S(h)$ relationship of Fig. 1 is replaced by a concave-like one in Fig. 3, the outflow hydrograph shape changes significantly (Fig. 4).

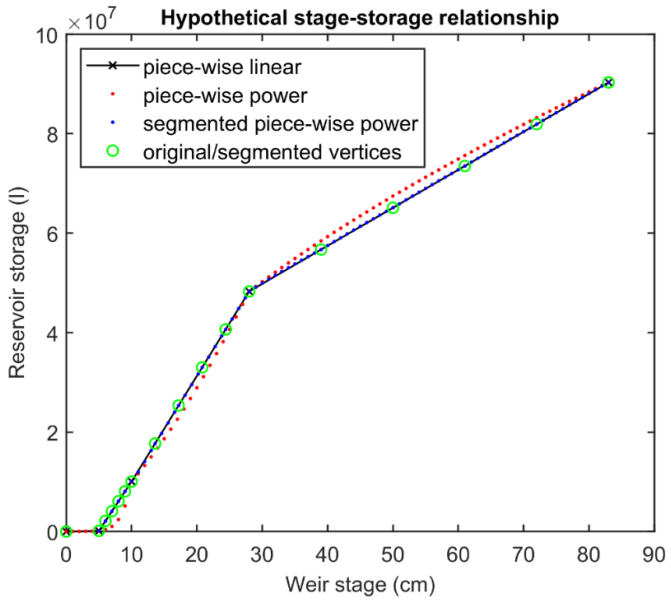

Fig. 3. Same as Fig. 1 but for a different stage-storage relationship.

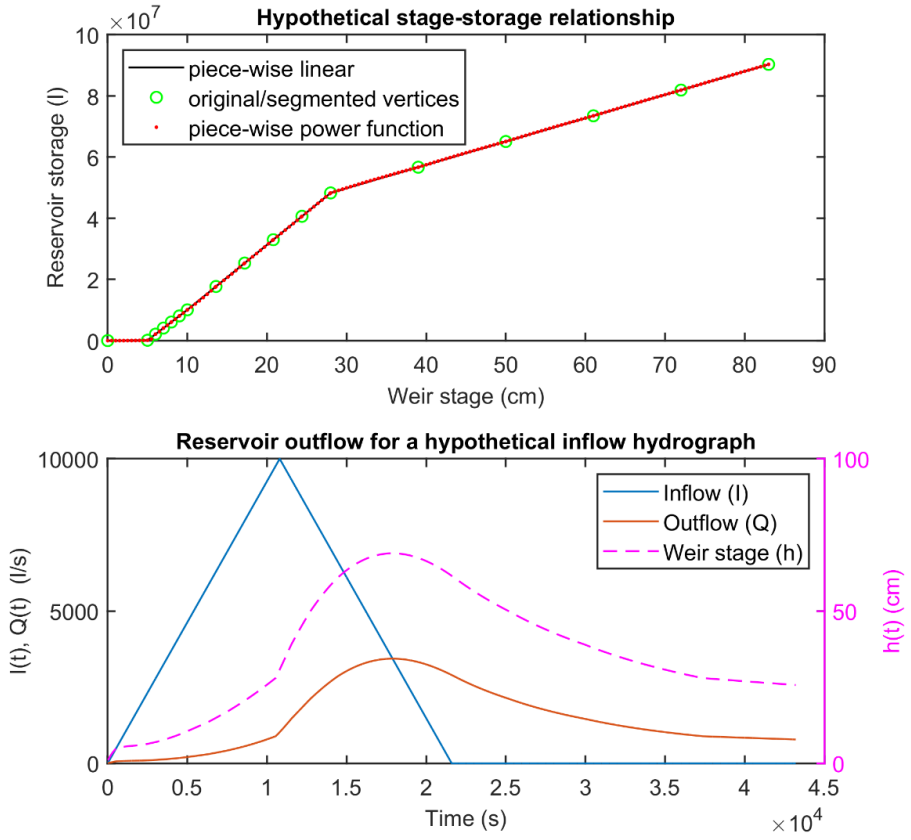

Fig. 4. Time series of design inflow as well as reservoir outflow and weir stage values. Maximum weir stage is 68.95 cm.

In order to facilitate the implementation of the required changes to the recursive application of Eq. (2) the commented MATLAB codes employed for the relevant calculations are included in the Appendix.

4 Conclusions

The recursive analytical solution [1] of the nonlinear storage equation can be used with realistic (piece-wise linear) stage-storage relationships after some minor modifications. The required changes involve segmentation of the original $S(h)$ line sections into as many smaller sections that a prescribed accuracy is met between the segmented line sections and the corresponding piece-wise power function approximations. When Eq. (2) is applied one must keep track of the outflow stage in order to employ the corresponding parameters of the piecewise power functions.

Acknowledgments

This study was supported by i) the Szechenyi Plan Plus program RRF-2.3.1-21-2022-00008; ii) FFT NP FTA of the Sustainable Development & Technologies National Programme of the Hungarian Academy of Sciences, and; iii) National Research, Development and Innovation Office of Hungary (NKFIH) under the National Research Excellence Programme – HIGHLIGHT_25, Project No. 152510.

Appendix

The first program code (piece_wise_S_h_curves.m) defines the vertices (h_i, S_i) of the stage-storage relationship and fits power-functions to the corresponding line sections. Each line section where a prescribed accuracy is not met is further segmented into a preset number of smaller line sections. The resulting vertices and power-function parameters are then saved.

The second program (nonlinSQ.m) reads in the segmented vertex coordinates and the corresponding power-function parameters and applies the piece-wise analytical solution, Eq. (2), of the nonlinear storage equation. The storage coefficient, k, is updated for each dt time increment by keeping track of which stage interval (h_i, h_{i+1}) the actual stage value, $h(t)$, falls into.

piece_wise_S_h_curves.m

```
clear all
clf

n=5;   %# of original vertices
mn=5;  %# of line segments between two vertices when power-function
       %is inaccurate
dx=1;  %increment in h for evaluating the piece-wise power functions
       %between vertices
dxx=1; %increment in h for evaluating the piece-wise power functions
       %over the new line segments
xmax=100;
ymax=10000;
yn=2;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%These lines to be replaced by the selected piece-wise S(h) curve
% x(1)=0.000001;y(1)=0.00000001;    %starting point for the S(h) curve
% x(2:n)=randi(xmax,n-1,1);         %n-1 integer numbers between 1 & xmax
% y(2:n)=randi(ymax,n-1,1);y=y.^yn; %same plus raising the y values to yn
% xsort=sort(x);
% ysort=sort(y);

% xsort=[0.000001 21 31 59 63];     %Figure 1
% ysort=[0.00000001 3794704 5313025 22184100 71301136];

xsort=[0.000001 5 10 28 83];        %Figure 3
ysort=[0.00000001 119025 10055241 48288601 90307009];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dsc=(ysort(end)-ysort(1))/100; %critical value for the piece-wise power
                               %function accuracy for segmentation

p1=plot(xsort,ysort,'kx-');    %plots the piece-wise linear S(h) function
title('Hypothetical stage-storage relationship')
xlabel('Weir stage (cm)')
ylabel('Reservoir storage (l)')
hold on

talal=0;j=0;
for i=1:n-1
    m(i)=(log(ysort(i+1))-log(ysort(i)))/... %fits the piece-wise
         (log(xsort(i+1))-log(xsort(i)));    %power functions
    a(i)=ysort(i+1)/xsort(i+1)^m(i);
    xx=xsort(i):dx:xsort(i+1);
    yy=a(i).*xx.^m(i);                       %plots them
    p2=plot(xx,yy,'r.');
    sl=(ysort(i+1)-ysort(i))/(xsort(i+1)-xsort(i));
    blin=ysort(i+1)-sl*xsort(i+1);
    yylin=sl.*xx+blin;

    if max(abs(yy-yylin))>dsc        %segmentation for inaccurate sections
       dxsur=(xsort(i+1)-xsort(i))/mn;
       xsortsur=xsort(i):dxsur:xsort(i+1);
       ysortsur=sl.*xsortsur+blin;
       nn=length(xsortsur);
       for ii=1:nn-1
           mm(ii)=(log(ysortsur(ii+1))-log(ysortsur(ii)))/...
                  (log(xsortsur(ii+1))-log(xsortsur(ii)));
           aa(ii)=ysortsur(ii+1)/xsortsur(ii+1)^mm(ii);
           xxx=xsortsur(ii):dxx:xsortsur(ii+1);
           yyy=aa(ii).*xxx.^mm(ii);
           p3=plot(xxx,yyy,'b.');
       end
       talal=1;
    end
    if talal>0
       for iii=1:length(xsortsur) %collects the segmented vertices'
           j=j+1; %coordinates into xall & yall & the corresponding a & m
           xall(j)=xsortsur(iii);yall(j)=ysortsur(iii); %into aall & mall
           if iii<length(xsortsur)
              mall(j)=mm(iii);aall(j)=aa(iii);
           end
       end
       talal=0;j=j-1;
       clear xx xxx yy yyy yylin
    else
       j=j+1;
       xall(j)=xsort(i);yall(j)=ysort(i);mall(j)=m(i);aall(j)=a(i); %for
    end                  %original line sections not needing segmentation

end
if xall(end)<xsort(end)
   j=j+1;
   xall(j)=xsort(end);yall(j)=ysort(end);
end
mall                               %the exponents of the piece-wise power
                                   %functions (original/segmented)

p4=plot(xall,yall,'go');           %checks the correctness of segmentation
```

```
legend([p1 p2 p3 p4],'piece-wise linear','piece-wise power',...
'segmented piece-wise power','original/segmented vertices','Location',...
'northwest')

save('xyam2', 'xall', 'yall', 'aall', 'mall'); %saves the
                  %original/segmented vertices and the fitted a & m values
```

nonlinSQ.m

```
clear all
clf

beadat=load('xyam2.mat');  %Reads in the segmented S(h) vertices & the
                           %corresponding a & m values
ai=beadat.aall;
mi=beadat.mall;
htp=beadat.xall;
Stp=beadat.yall;
n=length(ai);
ntp=length(htp);

subplot(2,1,1), plot(htp,Stp,'k-') %Plots the piece-wise linear &
                                   %power-function S(h) curves
hold on
subplot(2,1,1), plot(htp,Stp,'go')
dx=.5;
for i=1:n
    xmin=htp(i);xmax=htp(i+1);
    x=xmin:dx:xmax;
    y=ai(i).*x.^mi(i);
    subplot(2,1,1), plot(x,y,'r.')
end
title('Hypothetical stage-storage relationship')
xlabel('Weir stage (cm)')
ylabel('Reservoir storage (l)')
legend('piece-wise linear','original/segmented vertices',...
      'piece-wise power function','Location','northwest')

dt=10;                             %time-step of calculations in seconds
T=6*3600;                          %seconds (6 hours)
Qinmax=10000;                      %Maximum inflow value in l/s
Tmax=2*T;
Qt=zeros(1,Tmax/dt);               %The number of values scales with dt
t=dt:dt:Tmax;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Triangular inflow
sl=Qinmax/((T/2)/dt);              %slope of inflow
for i=1:(T/2)/dt
    Qe(i)=sl*i;
end
for i=(T/2)/dt+1:T/dt
    Qe(i)=Qinmax-sl*(i-(T/2)/dt);
end
for i=T/dt+1:Tmax/dt
    Qe(i)=0;
end

n=1.5;                             %Weir equation coeffs.
c=6;

%Initiating the m & a values plus related parameters for t=0 ############
Qo=0.1;                  %Qt (outflow) at the start of calculations (t=0);
ho=(Qo/c)^(1/n);         %Corresponding weir stage, h(t=0)
for i=1:ntp-1
    if ho>=htp(i) && ho<htp(i+1)
       m=mi(i);a=ai(i);
    end
end
nu=m/n;Kr=a/c^(nu);ff=1/(nu*Kr);k=ff/Qo^(nu-1);x=Qo;Qeo=0; %No inflow (Qe)
                                                           %yet at t=0
%#########################################################################

%Calculation of the first outflow Qt(1) at t=dt ########################
Fi=exp(-k*dt);
```

```
ga1=gammainc(k*dt,1)*(1/(k*dt)-exp(-k*dt)/gammainc(k*dt,1));
                                %gammainc(kdt,1)is the same as 1-exp(-kdt)
ga2=gammainc(k*dt,1)*(1-1/(k*dt)+exp(-k*dt)/gammainc(k*dt,1));
x=Fi*x+ga1*Qeo+ga2*Qe(1);       %Eq. (2)
Qt(1)=x;
ht(1)=(Qt(1)/c)^(1/n);
%Updating the m & a values plus related parameters for t=dt
for ii=1:ntp-1                  %Finding the relevant a & m value pairs
    if ht(1)>=htp(ii) && ht(1)<htp(ii+1)
       m=mi(ii);a=ai(ii);
    end
end
nu=m/n;Kr=a/c^(nu);ff=1/(nu*Kr);k=ff/Qt(1)^(nu-1);st(1)=a*(Qt(1)/c)^nu;
                                %st is storage
%#########################################################################

%Performing the previous two tasks for the remaining dt time-steps #####
for i=1:Tmax/dt-1
    Fi=exp(-k*dt);
    ga1=gammainc(k*dt,1)*(1/(k*dt)-exp(-k*dt)/gammainc(k*dt,1));
    ga2=gammainc(k*dt,1)*(1-1/(k*dt)+exp(-k*dt)/gammainc(k*dt,1));
    x=Fi*x+ga1*Qe(i)+ga2*Qe(i+1);
    Qt(i+1)=x;
    ht(i+1)=(Qt(i+1)/c)^(1/n);
    for ii=1:ntp-1              %Finding the relevant a & m value pairs
        if ht(i+1)>=htp(ii) && ht(i+1)<htp(ii+1)
           m=mi(ii);a=ai(ii);
        end
    end
    nu=m/n;Kr=a/c^(nu);ff=1/(nu*Kr);k=ff/Qt(i+1)^(nu-1);
    st(i+1)=a*(Qt(i+1)/c)^nu;
end
%#########################################################################

subplot(2,1,2), [hAx,hLine1,hLine2]=plotyy([t',t'],[Qe',Qt'],t,ht);
                                %Plotting time-series
hLine2.Color = 'm';
hLine2.LineStyle = '--';
title('Reservoir outflow for a hypothetical inflow hydrograph')
xlabel('Time (s)')
ylabel(hAx(1),'I(t), Q(t)  (l/s)') %left y-axis
ylabel(hAx(2),'h(t) (cm)')         %right y-axis
legend('Inflow (I)','Outflow (Q)','Weir stage (h)')
hAx(2).YColor = 'm';

Qtmax=max(Qt)                      %Maximum outflow rate (l/s)
htmax=(Qtmax/c)^(1/n)              %Maximum stage (cm)
Smax=max(st)                       %Maximum storage (cubic liter)

return
```

References

[1] Szilágyi, J. “A universal recursive analytical solution of the nonlinear storage equation for flood control reservoir sizing”. *Periodica Polytechnica Civil Engineering*. 70(1), pp. 151–155. 2026. https://doi.org/10.3311/PPci.42948

[2] Pirone, D., Cimorelli, L., D’Aniello, A., Pianese, D. “A novel generalized semi-analytical approach for flood control reservoir design”. *Water Resources Research*. 61, e2024WR039368. 2025. https://doi.org/10.1029/2024WR039368

[3] Szilágyi, J., Szöllősi-Nagy, A. “*Recursive Streamflow Forecasting: A State-Space Approach*”. Taylor & Francis, pp. 212. 2010. ISBN-13: 978-0-415-56901-9

[4] Szilágyi, J. “State-space discretization of the KMN-cascade in a sample-data system framework for streamflow forecasting”. *Journal of Hydrologic Engineering*. 8(6), pp. 339-347. 2003. https://doi.org/10.1061/(ASCE)1084-0699(2003)8:6(339)