Supplementary Materials: Entropy Parameter M in Modeling a Flow Duration Curve

Yu Zhang, Vijay P. Singh and Aaron R. Byrd

1. Data Availability

Daily maximum, minimum, and mean discharges, and reservoir and gauge station information were collected from the USGS website (https://waterdata.usgs.gov/nwis).

2. Code and Software

2.1. Single M Calculation

Numerical solutions for calculating M in this paper were conducted by using MATLAB R2016a.

According to the paper, the Lagrange multiplier λ_1 has the relationship between flow characteristics as shown below:

$$-\frac{1}{\lambda_1}-\frac{\mathrm{Q_{min}}\exp(-\lambda_1\mathrm{Q_{min}})-\mathrm{Q_{max}}\exp(-\lambda_1\mathrm{Q_{max}})}{\exp(-\lambda_1\mathrm{Q_{min}})-\exp(-\lambda_1\mathrm{Q_{max}})}=-\overline{\mathrm{Q}}$$

We can see that this is a non-linear equation of λ_1, so we need to find the numerical solution. In order to do this, we used MATLAB built-in functions. The 'fsolve' function is a nonlinear system solver which solves a problem specified by F(x)=0 for x. where F(x) is a function that returns a vector value. Here we make:

$$\mathrm{F}(\lambda 1)=-\frac{1}{\lambda_1}-\frac{\mathrm{Q_{min}}\exp(-\lambda_1\mathrm{Q_{min}})-\mathrm{Q_{max}}\exp(-\lambda_1\mathrm{Q_{max}})}{\exp(-\lambda_1\mathrm{Q_{min}})-\exp(-\lambda_1\mathrm{Q_{max}})}+\overline{\mathrm{Q}}$$

Then solve it in MATLAB.

The syntax is as follows: (use station 08093100):

```
filename='08093100.xlsx'%define the Excel file
sheet=1
xlRange1='E1:E49'
xlRange2='F1:F49'
xlRange3='G1:G49'
Qmax=xlsread(filename,sheet,xlRange1);%use prepared data in Excel
Qmin=xlsread(filename,sheet,xlRange2);
Qmean=xlsread(filename,sheet,xlRange3);
%make a for loop to calculate 49 years M values.
for i=1:49
    y=@(x)Qmean(i)-1/x-(Qmin(i)*exp(-x*Qmin(i))-Qmax(i)*exp(-x*Qmax(i)))/(exp(-x*Qmin(i))-exp(-x*Qmax(i)));
    sol(i)=fsolve(y,0.1);
end
sol
```

After getting λ_1, $M=Q_{max}*\lambda_1$

2.2. Apply Average M to Multiple Years FDC

All the files needed are shown below:

originData.m; sortData.m; calculateData.m; chooseData.m; simulateFQT.m; simulateFDC.m; runSimulation.m; runAll.m; simulateFDCByMean.m; runAllByMean.m; hydrologicYear.m

Step 1:

Copy data and discharge data to a csv file, then double-click it in MATLAB, choose numerical matrix, import data, here named data2.

Step 2:

Run sortData.m by entering sortData(data2)

Step 3:

Run runAll.m by entering [p R]runAll()

Step 4:

Run runAllByMean.m by entering [r2 sd]runAllByMean(p)

Step 5:

Import data of hydroYear as matrics using step1. Run hydrologicYear.m by entering [standardE R2]=hydrologicYear('data1994.mat',hydroYear)

Remember to change lines 8 and 9 when calculating different years of data.

Syntax for each m file are shown below:

originData.m

```
function M = originData(data, year)
%%data = sort(da,1,'ascend');
M= [];
aux = year*10000;
aux2 = (year+1)*10000;
isLastOne = 0;
i=1;
while(data(i,1)>aux2)
    i=i+1;
end
[f,c]= size(data);
while(isLastOne==0 && i<f)
if(data(i,1)>=aux && data(i,1)<aux2)
  M=[M;data(i,:)];
else
    isLastOne=1;
end
```

```
i=i+1;
end
```

sortData.m

```
function sortData(da)
%create all the files seprately.This is the fisrt step need to do
%da means the csv data name, enter sortData(data2) and run
%     Detailed explanation goes here
for(i=1965:2016)
    Batch = originData(da,i);
    [B,I] = sort(Batch(:,2),1,'ascend');
    data = Batch(I,:);
    fileName = sprintf('data%d.mat',i)
    save(fileName,'data');
end
end
```

calculateData.m

```
function data = calculateData( fileName )
%contains discharge,CDF,t/T named 'prepData' in the following files
%     Detailed explanation goes here
X = load(fileName);
da = X.data;
data=[1,da(1,2)];
[r,c]=size(da);
    for(i = 2:r)
        if(da(i,2)==data(i-1,2))
            data =[data; data(i-1,1) da(i,2)];
        else
            data = [data; data(i-1,1)+1 da(i,2)];
        end
    end
    data2 = [];
    for i = 1:r
        data2 = [data2; data(i,:), data(i,1)/data(r,1)];
    end
    data = [data2(1,:) r 1];
    for i=2:r
```

```
        if(data2(i,1) == data(i-1,1))
            data = [data; data2(i,:), data(i-1,4) data(i-1,5)];
        else
            data = [data; data2(i,:), (r-i+1) (r-i+1)/r];
        end
    end
end
```

chooseData.m

```
function data = chooseData( rawData, t1 )
%SCRIPT4 defining the data for simulation of the F(q) with t/T
%run by: t1=36:36:360; script4(script3(filename),t1
%it will return a matrics named 'ans'
%    Detailed explanation goes here

    data=[];
    r = length(rawData);
    l = length(t1);
    for i=1:l
        if(t1(i)<r)
            k = find(rawData(:,4)==t1(i));
            if(not(isempty(k)))
                if(length(k)>1)
                    v =t1(i)-length(k)+1;
                    data = [data;v,v/r,rawData(k(end),3)] ;
                end
                v =t1(i);
                data = [data;v,v/r,rawData(k(1),3)] ;
                %for h= length(k):-1:1
                %v =t1(i)-h+1;
                %data = [data;v,v/r,rawData(k(h),3)] ;
                %end
            else
                %disp('I COULDNT FIND THIS: ')
                %t1(i)
                %disp('I FOUND: ')
                %rawData(r-t1(i),4)
```

```
                k = find(rawData(:,4)==rawData(r-t1(i),4))
                if(not(isempty(k)))
                    %for h=length(k):-1:1
                    %disp('HEEEEEY');
                    if(length(k)>3)
                        v =rawData(k(end),4)-length(k)+1;
                        data = [data;v,v/r,rawData(k(end),3)] ;
                    end
                    % if(length(k)==3)
                    %     for j=3:-1:1
                    %         v = rawData(k(j),4)-j+1;
                    %         data = [data;v,v/r,rawData(k(j),3)] ;
                    %     end
                    % else
                    data = [data;t1(i),t1(i)/r,rawData(r-t1(i),3)];
                    % end

                    if(length(k)>2)
                        v =rawData(k(1),4);
                        data = [data;v,v/r,rawData(k(1),3)] ;
                    end
                    %end
                end
            end
        end
    end
    data = [data;r,r/r,rawData(1,3)];
    %%plot(data(:,2),data(:,3),'o');
end
```

simulateFQT.m

```
function p=simulateFQT( data, fileName )
%SCRIPT5 simulation process of F(q) with t/T
%run by: script5(ans) where 'ans' got by script4
%     Detailed explanation goes here
CDF = data(:,3);
```

```
T = data(:,2);
F = [ones(size(CDF)) log(T)];
ptemp = F\log(1-CDF)
p0 = [exp(ptemp(1)) ptemp(2)]
E = @(p) norm(CDF-1+p(1)*T.^p(2))^2;
%Minimize the error with fminsearch
[p ssr]=fminsearch(E,p0)
q = length(CDF)-length(p);
sd = sqrt(ssr/q)
close all
fig = figure;
scatter(T,CDF)
hold on
plot(T,1-p(1)*T.^p(2))
xlabel('t/T')
ylabel('F(Q)')
legend({'Observation','Simulation'},'location','northeast')
%annotation('textbox',[0.7 0.65 0.3 0.15],'String',{['a=' num2str(p(1))],['b=' num2str(p(2))],['sd='
num2str(sd)]},'LineStyle','none')
print(fig,strcat(fileName,'FQT.png'),'-dpng')
end
```

simulateFDC.m

```
function R=simulateFDC(p,prepData,fileName)
% filename='FDC_estimation.xlsx'%define the Excel file
% sheet1=29;
% xlRange3='B2:B366'
% xlRange4='E2:E366'
% Q=xlsread(filename,sheet1,xlRange3);
% T=xlsread(filename,sheet1,xlRange4);
%p=script6('data2011.mat')
%table2=script3('data2011.mat');
Q=prepData(:,2);
Qmax1=prepData(end,2);
Qmin1=prepData(1,2);
T=prepData(:,5);
% Qmax2=31506.18;
% Qmin2=56.30;
```

```
% Qmax3=91432.89;
% Qmin3=163.38;
M1=9.88;
% %STEP 3:use M from calculation due to principle of maximum entropy.
Q1=Qmax1*(-1/M1)*log(exp(-M1)-(exp(-M1)-exp(-M1*Qmin1/Qmax1))*p(1)*T.^p(2));
% Q2=Qmax2*(-1/M1)*log(exp(-M1)-(exp(-M1)-exp(-M1*Qmin2/Qmax2))*1.031*T.^0.781);
% Q3=Qmax3*(-1/M1)*log(exp(-M1)-(exp(-M1)-exp(-M1*Qmin3/Qmax3))*1.031*T.^0.781);
sd1=sqrt(norm(Q1-Q)^2/(length(Q)-1))
close all
fig = figure;
plot(T,Q)%figure of observed data
hold on
plot(T,Q1)%figure of pome result
% hold on
% plot(T,Q2,'--r')
% hold on
% plot(T,Q3,'--r')
xlabel('t/T')
ylabel('Q')
legend({'Observation','Simulation'},'location','northeast')
%annotation('textbox',[0.7 0.6 0.3 0.15],'String',{['sd=' num2str(sd1)]},'LineStyle','none')
print(fig,strcat(fileName,'FDC.png'),'-dpng')
R=corrcoef(Q1,Q);
end
```

runSimulattion.m

```
function [p R] = runSimulation(fileName)
t1 = 36:36:366;
prepData= calculateData(fileName);
choosenData = chooseData(prepData,t1);
p = simulateFQT(choosenData,fileName);
R= simulateFDC(p,prepData,fileName);
%R= simulateFDCByMean(p,prepData,fileName);%change it back when using other stations
end
```

runAll.m

```
function [p R]=runAll()
```

```
listing = dir('*.mat');
p=[];
R=[];
for(i=1:length(listing))
    listing(i).name
    [p1 R1]=runSimulation(listing(i).name);
    p=[p;p1];
    R=[R;R1];
end

end
```

simulateFDCByMean.m

```
function [R sd1]=simulateFDCByMean(pMean,prepData,fileName)
Q=prepData(:,2);
Qmax1=prepData(end,2);
Qmin1=prepData(1,2);
T=prepData(:,5);
M1=14.23;
% %STEP 3:use M from calculation due to principle of maximum entropy.
pMean
Q1=Qmax1*(-1/M1)*log(exp(-M1)-(exp(-M1)-exp(-M1*Qmin1/Qmax1))*pMean(1)*T.^pMean(2));
sd1=sqrt(norm(Q1-Q)^2/(length(Q)-1))
close all
fig = figure;
plot(T,Q)%figure of observed data
hold on
plot(T,Q1)%figure of pome result
xlabel('t/T')
ylabel('Q')
legend({'Observation','Simulation'},'location','northeast')
annotation('textbox',[0.7 0.6 0.3 0.15],'String',{['sd=' num2str(sd1)]},'LineStyle','none')
print(fig,strcat(fileName,'FDC.png'),'-dpng')
R=corrcoef(Q1,Q);
end
```

runAllByMean.m

```
function [r2 sd]=runAllByMean(p)
```

```
listing = dir('*.mat');
R=[];
sd1=[]
p
pMean = mean(p);
for(i=1:length(listing))
    listing(i).name
    [R1 sd2]=runSimulationByMean(pMean,listing(i).name);
    R=[R;R1];
    sd1=[sd1;sd2];
end
r2=R(1:2:end,2);
sd=sd1(:,1);
end
```

hydrologicYear.m

```
%run for 3 hydrologic years
%discharge:original Q:simulated
function [standardE R2]=hydrologicYear(fileName,hydroYear)%fileName=data1994.mat
q=calculateData(fileName);
discharge=q(:,2);
T=q(:,5);
for i=1:3
    Qmin(i)=hydroYear((i-1)*3+2,2);%colum3 means 1994,2=2009,1=2003
    Qmax(i)=hydroYear((i-1)*3+1,2);
end
    Qmin
    Qmax
M1=9.88;
Q1=Qmax(1)*(-1/M1)*log(exp(-M1)-(exp(-M1)-exp(-M1*Qmin(1)/Qmax(1)))*1.013*T.^0.886)
Q2=Qmax(2)*(-1/M1)*log(exp(-M1)-(exp(-M1)-exp(-M1*Qmin(2)/Qmax(2)))*1.013*T.^0.886)
Q3=Qmax(3)*(-1/M1)*log(exp(-M1)-(exp(-M1)-exp(-M1*Qmin(3)/Qmax(3)))*1.013*T.^0.886)
standardE=sqrt(norm(Q1-discharge)^2/(length(discharge)-1))
figure
plot(T,discharge)%figure of observed data
hold on
```

```
plot(T,Q1)%figure of pome result
hold on
plot(T,Q2,'--r')%figure of interval
hold on
plot(T,Q3,'--r')%figure of interval
xlabel('t/T')
ylabel('Q')
legend({'Observation','Simulation','95% interval'},'location','northeast')
%annotation('textbox',[0.7 0.6 0.3 0.15],'String',{['sd=' num2str(standardE)]},'LineStyle','none')
R2=corrcoef(Q1,discharge)
end
```

2.3. Gamma Fit for Probability Distribution

In this paper, we chose Gamma distribution to fit for the probability distribution of the maximum, minimum, and mean discharges. The process was also conducted in MATLAB. The syntax is shown below:

```
filename='example08093100.xlsx'
sheet1=1;
xlRange1='C2:C67';
y1 =xlsread(filename,sheet1,xlRange1);
%PD=fitdist(y,'burr')
[phat,pci] = mle(y1,'distribution','gamma')
pcdf=cdf('gamma',sortrows(y1),1.341,17.857);
figure
plot(sortrows(y1),pcdf)
title('minimum flow data')
xlabel('minimum flow')
ylabel('cdf')
figure
histfit(y1,7,'gamma')
title('Histogram of minimum flow data with a Gamma distribution fit')
xlabel('minimum flow')
ylabel('count')
```