

水文预报课程设计

褒河流域



作者：刘彦龙

指导教师：程磊

专业名称：水文与水资源工程

2018-11月

目 录

[一、 概述 3](#_Toc529699457)

[1.1 流域概况 3](#_Toc529699458)

[1.2 流域水文资料 3](#_Toc529699459)

[1.2.1 褒河流域水文资料的整理 3](#_Toc529699460)

[1.3 任务要求 5](#_Toc529699461)

[二、 流域产流方案 5](#_Toc529699462)

[2.1 流程与方案 5](#_Toc529699463)

[2.2 流域蒸散发 6](#_Toc529699464)

[2.3 蓄满产流 8](#_Toc529699465)

[2.3.1 基本原理 8](#_Toc529699466)

[2.3.2蓄水容量曲线 8](#_Toc529699467)

[2.3.2 降水产流量计算 9](#_Toc529699468)

[2.4 三水源划分 10](#_Toc529699469)

[2.4.1 自由水的蓄水流量曲线 10](#_Toc529699470)

[2.4.2 流量计算 11](#_Toc529699471)

[三、 流域汇流方案 12](#_Toc529699472)

[3.1 概述 12](#_Toc529699473)

[3.2 汇流划分 12](#_Toc529699474)

[3.2.1 地面径流汇流 12](#_Toc529699475)

[3.2.2 壤中流汇流 13](#_Toc529699476)

[3.2.3 地下径流汇流 13](#_Toc529699477)

[四、 预报方案检验 14](#_Toc529699478)

[4.1 概述 14](#_Toc529699479)

[4.2 精度评定 14](#_Toc529699480)

[4.2.1 误差指标 14](#_Toc529699481)

[4.2.2 许可误差 15](#_Toc529699482)

[4.2.3 预报项目精度评定 15](#_Toc529699483)

[4.3 模型精度评定结果分析 16](#_Toc529699484)

[4.3.1 率定期评定 16](#_Toc529699485)

[4.3.2 检验期评定 17](#_Toc529699486)

[五、 存在的问题与改进方案 20](#_Toc529699487)

[参考文献 21](#_Toc529699488)

[结语 21](#_Toc529699489)

[附件 22](#_Toc529699490)

[1. 计算代码 22](#_Toc529699491)

**图表目录**

[表格 1 褒河流域次洪统计 3](#_Toc529699492)

[表格 2 马道站次洪水统计表 5](#_Toc529699493)

[表格 3 褒河流域三水源蓄满产流模型产流参数 10](#_Toc529699494)

[表格 4 褒河流域分水源汇流参数 14](#_Toc529699495)

[表格 5 预报项目等级精度表 15](#_Toc529699496)

[表格 6 精度评定 16](#_Toc529699497)

[表格 7 检验期预报精度 17](#_Toc529699498)

[表格 8 检验期峰现时间精度 18](#_Toc529699499)

[图2. 1 包气带蓄水容量曲线 8](#_Toc529699500)

[图2. 2 流域初始土湿分布于降雨产流量 9](#_Toc529699501)

[图2. 3 自由水蓄水容量曲线 10](#_Toc529699502)

2018年11月11日

[图4. 1 降雨洪水拟合过程 19](#_Toc529699503)

[图4. 2 870718号洪水过程拟合 19](#_Toc529699504)

[图4. 3 900516号洪水过程拟合 20](#_Toc529699505)

[图4. 4 效果较差的拟合 20](#_Toc529699506)

1. 概述
   1. 流域概况

褒河流域在陕西省南部汉江上游，经度106°48′15″—107°25′34″，纬度33°38′03″—34°11′08″，流域集水面积约为3415 km2，海拔900～3400m。褒河流域雨量充沛、洪水频繁，峰高量大，研究其出口断面洪水预报方案对下游人民生命财产安全起着十分重要的作用。

* 1. 流域水文资料
     1. 褒河流域水文资料的整理

马道水文站控制集水面积约3415km2，流域内有太白、大坪里等13个雨量站，各站均有较长期的实测雨量资料（1980~1990）。在洪水预报方案的研制中，已经将1980～1986年资料作为模型率定期，选出了22场暴雨与褒河同期实测流量资料作为优选产流参数和汇流过程的依据。现需要将预留的1987~1990年资料进行整理，为模型产汇流方案检验提供场次洪水资料。资料的整理和数据文件的组织都已通过编制的软件由计算机完成。1980～1986年划分和计算的场次暴雨洪水结果如表1所示，表1中1987~1990年的为待求值。

①暴雨洪水场次的划分

1987~1990年洪水与暴雨场次的大致划分如表2所示。

②场次洪水径流深的推求

褒河流域退水曲线中蓄泄系数K=48小时。

表格 1 褒河流域次洪统计

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 洪号 | 降雨P | 实测RO | 计算RC | 绝对误差 | 相对误差 |
| 1 | 800616 | 76.3 | 22.4 | 16.6 | -5.8 | -25.9 |
| 2 | 800703 | 164.1 | 104.2 | 80.5 | -23.7 | -22.7 |
| 3 | 800803 | 55.2 | 30.1 | 27.3 | -2.8 | -9.3 |
| 4 | 800915 | 78.3 | 44.5 | 39.4 | -5.1 | -11.5 |
| 5 | 810714 | 119.6 | 72.7 | 67.3 | -5.4 | -7.4 |
| 6 | 810819 | 466.5 | 404.3 | 394.7 | -9.6 | -2.4 |
| 7 | 811001 | 59.5 | 41.2 | 30.7 | -10.5 | -25.5 |
| 8 | 820830 | 124.8 | 46.3 | 60.2 | 13.9 | 30 |
| 9 | 820926 | 65.5 | 21.5 | 27.0 | 5.5 | 25.6 |
| 10 | 830624 | 79.8 | 35.8 | 32.2 | -3.6 | -10.1 |
| 11 | 830721 | 105.0 | 40.0 | 50.1 | 10.1 | 25.3 |
| 12 | 830819 | 64.5 | 28.4 | 25.8 | -2.6 | -9.2 |
| 13 | 830907 | 85.3 | 39.0 | 36.8 | -2.2 | -5.6 |
| 14 | 840624 | 60.8 | 21.0 | 22.9 | 1.9 | 9 |
| 15 | 840706 | 58.9 | 28.9 | 27.2 | -1.7 | -5.9 |
| 16 | 840804 | 49.9 | 19.2 | 20.7 | 1.5 | 7.8 |
| 17 | 840905 | 134.0 | 71.5 | 84.6 | 13.1 | 18.3 |
| 18 | 850517 | 24.5 | 13.1 | 10.4 | -2.7 | -20.6 |
| 19 | 850711 | 50.6 | 15.2 | 16.9 | 1.7 | 11.2 |
| 20 | 850914 | 56.1 | 28.6 | 23.6 | -5 | -17.5 |
| 21 | 860701 | 72.7 | 26.1 | 21.6 | -4.5 | -17.2 |
| 22 | 860710 | 41.8 | 18.0 | 15.4 | -2.6 | -14.4 |
| 23 | 870627 |  |  |  |  |  |
| 24 | 870718 |  |  |  |  |  |
| 25 | 870804 |  |  |  |  |  |
| 26 | 870903 |  |  |  |  |  |
| 27 | 880709 |  |  |  |  |  |
| 28 | 880818 |  |  |  |  |  |
| 29 | 880901 |  |  |  |  |  |
| 30 | 890428 |  |  |  |  |  |
| 31 | 890522 |  |  |  |  |  |
| 32 | 890707 |  |  |  |  |  |
| 33 | 890819 |  |  |  |  |  |
| 34 | 900516 |  |  |  |  |  |
| 35 | 900706 |  |  |  |  |  |
| 36 | 900816 |  |  |  |  |  |

表格 2 马道站次洪水统计表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NO | 洪号 | 流量起止时间 | 实测洪峰(m3/s) | 实测峰现时间 | 模拟洪峰(m3/s) | 模拟峰现时间 | 实测洪水总量(mm) | 模拟洪水总量(mm) |
| 1 | 870627 | 06.25-07.03 |  |  |  |  |  |  |
| 2 | 870718 | 07.16-07.26 |  |  |  |  |  |  |
| 3 | 870804 | 08.02-08.12 |  |  |  |  |  |  |
| 4 | 870903 | 08.31-09.08 |  |  |  |  |  |  |
| 5 | 880709 | 07.03-07.17 |  |  |  |  |  |  |
| 6 | 880818 | 08.17-08.25 |  |  |  |  |  |  |
| 7 | 880901 | 08.30-09.09 |  |  |  |  |  |  |
| 8 | 890428 | 04.26-05.06 |  |  |  |  |  |  |
| 9 | 890522 | 05.20-05.30 |  |  |  |  |  |  |
| 10 | 890707 | 07.03-07.15 |  |  |  |  |  |  |
| 11 | 890819 | 08.14-08.25 |  |  |  |  |  |  |
| 12 | 900516 | 05.13-05.26 |  |  |  |  |  |  |
| 13 | 900706 | 07.04-07.15 |  |  |  |  |  |  |
| 14 | 900816 | 08.11-08.27 |  |  |  |  |  |  |

* 1. 任务要求

本课程设计的主要任务是：根据建立的流域洪水预报方案，应用预留的水文资料，对此方案进行检验。具体工作内容如下：

①流域暴雨洪水场次资料的整理。

②流域产流方案的检验。

③流域汇流方案的检验。

④流域洪水预报方案的评定。

1. 流域产流方案

2.1 流程与方案

本方案采用蓄满产流模型，应用新安江模型。主要过程分为流域蒸散发的三层蒸发计算、蓄满产流计算、水源划分等三个部分。

2.2 流域蒸散发

对于长期的产流量估计，蒸发常是决定性因素。（流域蒸散发量很难由直接观测资料确定，一般通过模型计算得到。）

影响流域蒸散发的主要因素是土壤蒸发，而影响土壤蒸发的因素主要是土壤蒸发能力和土壤含水量。即：

(2. 1)

式中: ；

土壤蒸散发过程大体上可以划分为三个基本阶段：

a) 土壤含水量供水充分，只受气候因素影响，稳定蒸散发阶段。

b) 土壤含水量和气候同时影响，蒸散发与土壤含水量比例变化蒸散发阶段

c) 取决于地下水埋深与气候因素，常倍数深层蒸散发扩散阶段。

在本模型中，按土壤垂向分布的不均匀性将土层分为三层，用三层蒸散发模型计算蒸散发量。三层蒸发计算公式如下：

上层蒸发量： (2. 2)

下层蒸发量： (2. 3)

深层蒸发量： (2. 4)

总蒸发量： (2. 5)

式中：

其中流域蒸发能力与水面蒸发能力之间可粗略概化为如下线性关系：

(2. 6)

式中：

三层蒸发模式按照先上层后下层的次序，具体分如下四种情况计算:

1. 当时：

(2. 7)

1. 当时：

(2. 8)

(2. 9)

1. 当时：

(2. 10)

(2. 11)

1. 当时：

(2. 12)

(2. 13)

其中初始土壤湿度：

(2. 14)

(2. 15)

(2. 16)

式中：

新安江模型蒸发计算相关参数：

KC：为蒸散发能力折算系数，是指流域蒸散发能力与实测水面蒸发值之比。此参数控制着总水量平衡，因此，对水量计算是十分重要的。

WUM：为上层蓄水容量，它包括植物截留量。在植被与土壤比较发育的流域,约为20mm；在植被与土壤颇差的流域，约为5～6mm。

WLM：为下层蓄水容量。可取60～90mm。

WDM：为深层蓄水容量。由WM=WUM+WLM+WDM关系可以省去此参数的率定。

C：为深层蒸散发系数。它决定于深根植物占流域面积的比数，同时也与WUM+WLM值有关，此值越大，深层蒸散发越困难。一般经验，在江南湿润地区C值约为0.15～0.20左右，而在华北半湿润地区则在0.09～0.12左右。

FE：为初始土壤含水容量折算系数，即各层土壤水均为容量的FE倍：如，

2.3 蓄满产流

2.3.1 基本原理

蓄满产流模型的的基本原理可表达为：

(2. 17)

式中：

2.3.2蓄水容量曲线

流域蓄水容量曲线时将流域内各地点包气带的蓄水水容量，按从小到达顺序排列得到的一条蓄水容量与相应面积关系的统计曲线。如图2.1所示图中纵坐标WM’为各地点包气带蓄水容量值，WMM为其中最大值，一般都以mm表示，横坐标为面积的相对值f/F，F是全流域面积，f为流域内包气带蓄水容量小于或等于WM’的面积，曲线所围的面积WM为全流域平均蓄水容量。流域蓄水容量曲线反映了流域包气带缺水容量分布特征，如图2.1所示：

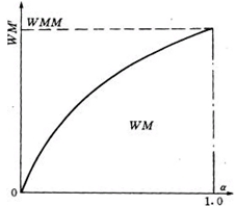


图2. 1 包气带蓄水容量曲线

据经验分析，蓄水容量曲线可由如下指数方程近似描述

(2. 18)

式中：B反映了流域包气带蓄水容量分布的不均匀性。B值越小表示越均匀。当B=0时表示流域内包气带蓄水容量均匀不变，而B值越大表示越不均匀。

据上式积分可得:

(2. 19)

式中：

2.3.2 降水产流量计算

一般情况下，降雨前的初始土壤含水量不为零，这时，初始土壤含水量在流域的分布直接影响降雨产流量值。降雨前的初始土壤含水量分布式不相同的，但从多次平均的统计角度，认为分布规律也符合图2.2的变化。

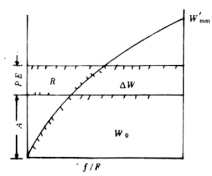


图2. 2 流域初始土湿分布于降雨产流量

根据图2.2，W计算式为

(2. 20)

积分得：

(2. 21)

对应α为：

(2. 22)

设扣除雨期蒸发后的降雨量为PE，则总径流量R的计算公式为

(2. 23)

当，即局部产流时：

(2. 24)

当，即全流域产流时：

(2. 25)

新安江模型蓄满产流相关参数：

WM：为流域蓄水容量，是流域干湿程度的指标。一般分为上层 WUM、下层WLM和深层WDM，约为120～180mm。

B：为蓄水容量曲线的指数。它反映流域上蓄水容量分布的不均匀性。一般经验，流域越大，各种地质地形配置越多样，B值也越大。在山丘区域，很小面积（几平方公里）的B值为0.1左右，中等面积（300平方公里以内）的B值为0.2～0.3左右，较大面积（数千平方公里）的B值为0.3～0.4左右。

IMP：不透水层占全流域面积

表格 3 褒河流域三水源蓄满产流模型产流参数

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 参数名 | WM | WUM | WLM | WDM | KC | C | B | IMP | FE |
| 参数值 | 115 | 20 | 25 | 70 | 1.04 | 0.16 | 1.75 | 0.054 | 0.8 |

注：KC为流域蒸发折算系数；FE为初始土壤含水容量折算系数，即各层土壤水均为容量的FE倍：如WUM0=FE\*WUM，S0=SM\*FE。

2.4 三水源划分

地面以下的径流由多种产流机制形成，在流域出口断面流量的退水过程线上常呈现水源三段退水的退水特征。这三段退水的径流主要成分分别为地面、壤中和地下三种水源。在新安江模型中常用水箱模型以解决水源划分问题。

2.4.1 自由水的蓄水流量曲线

自由水的蓄水容量曲线可以表示为图2.3所示。

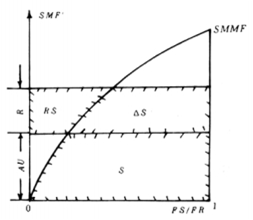


图2. 3 自由水蓄水容量曲线

据图2.3可得自由水的蓄水容量曲线为：

(2. 26)

因此：

(2. 27)

(2. 28)

式中： ；

；

；

；

；

；

2.4.2 流量计算

①  的产流计算

当时，即地面净雨为0，因此，但此时地下水的自由水库含水量不为零，因此壤中流，地下径流，则此时：

(2. 29)

(2. 30)

(2. 31)

(2. 32)

(2. 33)

② 的产流计算

当分两种情况：

1. 若，则：

(2. 34)

(2. 35)

(2. 36)

(2. 37)

1. 若，则：

(2. 38)

(2. 39)

(2. 40)

(2. 41)

式中：

；

；

；

；

；

；

其余变量意义和前面式子相同。

新安江模型水源划分相关参数：

SM:为流域平均自由水蓄水容量，本参数受降雨资料时段均化的影响，当以日为计算时段长时，一般流域的 SM 值约为 10～50mm，当所选取的计算时段长较小时，SM 要增大，这个参数对地面径流的多少起着决定性作用，因此十分重要。

EX :为自由水蓄水容量曲线指数，它表示自由水容量分布不均匀性。通常EX 取在 1～1.5 之间。

KI:为自由水蓄水库对壤中流的出流系数，KG 为自由水蓄水库对地下径流出流系数，这两个出流系数是并联的，其和代表着自由水出流的快慢。一般来说，

KI+KG＝0.7，相当于从雨止到壤中流止的时间为 3 天。本流域中选择对 KI 进行率定，KG 有以上关系式得出。

1. 流域汇流方案

3.1 概述

流域汇流计算包括坡地和河网两个汇流阶段。

新安江模型中流域汇流分：地面汇流、壤中流汇流和地下汇流。汇流参数的优选以流域出口模拟流量过程与实测流量过程拟合最优为目标函数。壤中流和地下汇流采用线性水库的方法。地面汇流采用时段地面经验单位线法，用实测雨洪资料分析的单位线方法。

3.2 汇流划分

根据三水源的划分，汇流过程也可以由三部分组成：地面径流的汇流、壤中流的汇流和地下水的汇流。

3.2.1 地面径流汇流

地面径流汇流采用Nash单位线法，利用单位线的倍比和叠加假定逐日进行计算。因所给单位线∆t=24h，与计算时段长相等，故不需转换，可直接带入计算。净雨量、出流量与UH纵坐标q之间的关系如下：

(3. 1)

式中： ；

；

;

；

；

(3. 2)

(3. 3)

3.2.2 壤中流汇流

表层自由水侧向流动，出流后成为表层壤中流进入河网，若土层较厚，表层自由水还可以渗入到深层土，经过深层土的调蓄作为才进入河网。壤中流汇流可采用线性水库或滞后演算法计算。当采用线性水库计算时，计算公式为：

(3. 4)

式中： ；

；

；

；

。

3.2.3 地下径流汇流

地下径流汇流采用线性水库，计算公式为：

(3. 5)

式中： ；

；

；

；

。

流域出口断面总流量为

(3. 6)

式中：；

；

；

；

新安江模型汇流计算参数：

KKSS：为壤中流水库的消退系数。如无深层壤中流时，KKI趋于零。当深层壤中流很丰富时，KKSS趋于0.9。相当于汇流时间为10天。

KKG：为地下水库的消退系数。如以日为计算时段长，此值一般为0.98～

0.998，相当于汇流时间为50～500天。

表格 4 褒河流域分水源汇流参数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 参数名 | SM | EX | KG | KKG | KSS | KKSS |
| 参数值 | 40 | 1.9 | 0.06 | 0.995 | 0.05 | 0.885 |

1. 预报方案检验

4.1 概述

预报方案效果评定和有效性检验，一般是将具有良好代表性的资料系列分为率定期和检验期。评定是采用率定期可利用的资料编制方案、估计参数、确定预报方案再进行模拟，通过模拟与实测水文要素之间的比较，分析预报方案在率定期的效果与有效性。检验则是采用检验期预报环境可利用资料，用预报方案进行模拟，通过模拟结果与实测水文要素之间的比较，分析预报方案在检验期的效果与有效性。由于预报误差出现是随机的，率定期和检验期的评定精度指标不会完全一致。

4.2 精度评定

洪水预报精度评定包括预报方案精度评定、作业预报的精度等级评定和预报时效等级评定等。评定项目主要有洪峰流量(水位)、洪峰出现时间、洪量(径流量)等。

4.2.1 误差指标

洪水预报的误差指标采用以下三种：

绝对误差：绝对误差=|预报值-实测值|

相对误差：相对误差 =( 预报误差 / 实测值) x 100%

确定性系数：

(4. 1)

式中： ；

；

；

；

。

4.2.2 许可误差

许可误差是依据预报精度的使用要求和实际预报技术水平等综合确定的误差允许范围，不同的预报方法和预报要素对许可误差的规定不同，对许可误差规定亦不同。

（1）洪峰预报许可误差

降雨径流预报以实测洪峰流量的 20%作为许可误差；河道流量（水位）预报以预见期内实测变幅的 20%作为许可误差。

（2）峰现时间预报许可误差

峰现时间以预报根据时间至实测洪峰出现时间之间时距的 30%作为许可误差，当许可误差小于 3Δt 或一个计算时段长，则以 3Δt 或一个计算时段长作为许可误差。

（3）径流深预报许可误差

径流深预报以实测值的 20%作为许可误差，当该值大于 20mm 时，为20mm；小于 3mm 时，取 3mm。

4.2.3 预报项目精度评定

(1)合格预报

一次预报的误差小于许可误差时，为合格预报。合格预报次数与预报总次数之比的百分数为合格率，表示多次预报总体的精度水平。合格率:

(4. 2)

式中： ；

；

。

(2)预报项目精度等级

预报项目的精度按合格率或确定性系数分为三个等级如表 5 所示：

表格 5 预报项目等级精度表

|  |  |  |  |
| --- | --- | --- | --- |
| 精度等级 | 甲 | 乙 | 丙 |
| 合格率(%) | QR>=85.0 | 85.0>QR>=70.0 | 70.0>QR>=60.0 |
| 确定性系数 | DC>0.90 | 0.90>=DC>=0.70 | 0.70>DC>=0.50 |

4.3 模型精度评定结果分析

根据 4.2 所述精度评定方法，对于本模型预报的径流深，洪峰流量，峰现时间，洪水预报过程进行评定。

4.3.1 率定期评定

通过计算，模型率定期预报总次数为 22，合格预报次数为 15，预报合格率为 68.18%，预报精度达到丙等。如表6。

表格 6 精度评定

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 率定期 | 序号 | 洪号 | 降雨P | 实测RO | 计算RC | 绝对误差 | 相对误差 | 是否合格 | R2 |
| 1 | 800616 | 76.3 | 22.4 | 16.6 | -5.8 | -25.9 | *0* |  |
| 2 | 800703 | 164.1 | 104.2 | 80.5 | -23.7 | -22.7 | *0* |  |
| 3 | 800803 | 55.2 | 30.1 | 27.3 | -2.8 | -9.3 | 1 |  |
| 4 | 800915 | 78.3 | 44.5 | 39.4 | -5.1 | -11.5 | 1 |  |
| 5 | 810714 | 119.6 | 72.7 | 67.3 | -5.4 | -7.4 | 1 |  |
| 6 | 810819 | 466.5 | 404.3 | 394.7 | -9.6 | -2.4 | 1 |  |
| 7 | 811001 | 59.5 | 41.2 | 30.7 | -10.5 | -25.5 | *0* |  |
| 8 | 820830 | 124.8 | 46.3 | 60.2 | 13.9 | 30 | *0* |  |
| 9 | 820926 | 65.5 | 21.5 | 27.0 | 5.5 | 25.6 | *0* |  |
| 10 | 830624 | 79.8 | 35.8 | 32.2 | -3.6 | -10.1 | 1 |  |
| 11 | 830721 | 105.0 | 40.0 | 50.1 | 10.1 | 25.3 | *0* |  |
| 12 | 830819 | 64.5 | 28.4 | 25.8 | -2.6 | -9.2 | 1 |  |
| 13 | 830907 | 85.3 | 39.0 | 36.8 | -2.2 | -5.6 | 1 |  |
| 14 | 840624 | 60.8 | 21.0 | 22.9 | 1.9 | 9 | 1 |  |
| 15 | 840706 | 58.9 | 28.9 | 27.2 | -1.7 | -5.9 | 1 |  |
| 16 | 840804 | 49.9 | 19.2 | 20.7 | 1.5 | 7.8 | 1 |  |
| 17 | 840905 | 134.0 | 71.5 | 84.6 | 13.1 | 18.3 | 1 |  |
| 18 | 850517 | 24.5 | 13.1 | 10.4 | -2.7 | -20.6 | *0* |  |
| 19 | 850711 | 50.6 | 15.2 | 16.9 | 1.7 | 11.2 | 1 |  |
| 20 | 850914 | 56.1 | 28.6 | 23.6 | -5 | -17.5 | 1 |  |
| 21 | 860701 | 72.7 | 26.1 | 21.6 | -4.5 | -17.2 | 1 |  |
| 22 | 860710 | 41.8 | 18.0 | 15.4 | -2.6 | -14.4 | 1 |  |
| 检验期 | 23 | 870627 | 53.7 | 19.1 | 18.6 | -0.6 | -3.0 | 1 | 80.34 |
| 24 | 870718 | 79.7 | 29.9 | 39.6 | 9.6 | 32.2 | 0 | 90.19 |
| 25 | 870804 | 116.8 | 41.4 | 61.1 | 19.8 | 47.8 | 0 | 42.07 |
| 26 | 870903 | 41.1 | 8.8 | 14.7 | 5.9 | 67.5 | 0 | -107.51 |
| 27 | 880709 | 144.9 | 50.4 | 62.1 | 11.7 | 23.2 | 0 | 84.02 |
| 28 | 880818 | 50.2 | 31.4 | 37.8 | 6.4 | 20.4 | 0 | 74.46 |
| 29 | 880901 | 55.6 | 24.6 | 33.0 | 8.4 | 34.0 | 0 | 72.03 |
| 30 | 890428 | 45.7 | 26.7 | 12.7 | -14.0 | -52.5 | 0 | -12.39 |
| 31 | 890522 | 27.5 | 18.0 | 12.0 | -6.0 | -33.3 | 0 | 37.23 |
| 32 | 890707 | 86 | 29.0 | 24.0 | -5.1 | -17.5 | 1 | 61.97 |
| 33 | 890819 | 129.5 | 43.6 | 57.6 | 14.0 | 32.1 | 0 | 56.51 |
| 34 | 900516 | 65.9 | 25.0 | 23.5 | -1.4 | -5.8 | 1 | 91.78 |
| 35 | 900706 | 156.3 | 100.5 | 115.3 | 14.8 | 14.7 | 1 | 71.62 |
| 36 | 900816 | 152.4 | 69.8 | 88.7 | 18.9 | 27.0 | 0 | 58.13 |

4.3.2 检验期评定

检验期确定性系数与预报等级计算结果如下所示，检验期共14场次洪水，其中相对误差在20%以内的共4场，值得注意的是，在十四场次洪水预报中有一场次确定性系数超过-100%，此项数据明显错误，但笔者花了很长时间也没有找到问题所在，又迫于时间压力以及本着不造假的原则，因此未做出改正。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO | 洪号 | 流量起止时间 | 确定性系数 | 等级 |
| 1 | 870627 | 06.25-07.03 | 0.80 | 乙 |
| 2 | 870718 | 07.16-07.26 | 0.90 |  |
| 3 | 870804 | 08.02-08.12 | 0.42 |  |
| 4 | 870903 | 08.31-09.08 | -1.08 |  |
| 5 | 880709 | 07.03-07.17 | 0.84 |  |
| 6 | 880818 | 08.17-08.25 | 0.74 |  |
| 7 | 880901 | 08.30-09.09 | 0.72 |  |
| 8 | 890428 | 04.26-05.06 | -0.12 |  |
| 9 | 890522 | 05.20-05.30 | 0.37 |  |
| 10 | 890707 | 07.03-07.15 | 0.62 | 丙 |
| 11 | 890819 | 08.14-08.25 | 0.57 |  |
| 12 | 900516 | 05.13-05.26 | 0.92 |  |
| 13 | 900706 | 07.04-07.15 | 0.72 | 甲 |
| 14 | 900816 | 08.11-08.27 | 0.58 | 乙 |

表格 7 检验期预报精度

峰现时间预报结果如下表所示，其中共14场次洪水，峰现时间预报正确13场，一场绝对误差为1，符合精度要求，总体合格率100%，因此洪峰预报效果较好。

表格 8 检验期峰现时间精度

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NO | 洪号 | 流量起止时  间 | 实测峰现  时间 | 模拟峰现  时间 | 绝对误差 | 是否合格 | 合格率 |
| 1 | 870627 | 06.25-07.03 | 6/27 | 6/27 | 0 | 1 | 100.00% |
| 2 | 870718 | 07.16-07.26 | 7/18 | 7/18 | 0 | 1 |
| 3 | 870804 | 08.02-08.12 | 8/4 | 8/4 | 0 | 1 |
| 4 | 870903 | 08.31-09.08 | 9/3 | 9/3 | 0 | 1 |
| 5 | 880709 | 07.03-07.17 | 7/9 | 7/9 | 0 | 1 |
| 6 | 880818 | 08.17-08.25 | 8/18 | 8/18 | 0 | 1 |
| 7 | 880901 | 08.30-09.09 | 9/1 | 9/2 | 1 | 1 |
| 8 | 890428 | 04.26-05.06 | 4/28 | 4/28 | 0 | 1 |
| 9 | 890522 | 05.20-05.30 | 5/22 | 5/22 | 0 | 1 |
| 10 | 890707 | 07.03-07.15 | 7/7 | 7/7 | 0 | 1 |
| 11 | 890819 | 08.14-08.25 | 8/19 | 8/19 | 0 | 1 |
| 12 | 900516 | 05.13-05.26 | 5/16 | 5/16 | 0 | 1 |
| 13 | 900706 | 07.04-07.15 | 7/6 | 7/6 | 0 | 1 |
| 14 | 900816 | 08.11-08.27 | 8/16 | 8/16 | 0 | 1 |

检验期模型中降雨、模拟径流与实测径流关系如下图4.1所示，其中上方蓝色柱状图为降雨过程，下方黄色折线图为模拟洪水过程，红色折线图为实测洪水过程。从图中可以明显的看出洪水过程相较于降雨过程的滞后性。从红色以及黄色的折线契合程度来看整体上符合洪水实际过程，但在上面的计算中可以知道细节部分存在一定的误差。

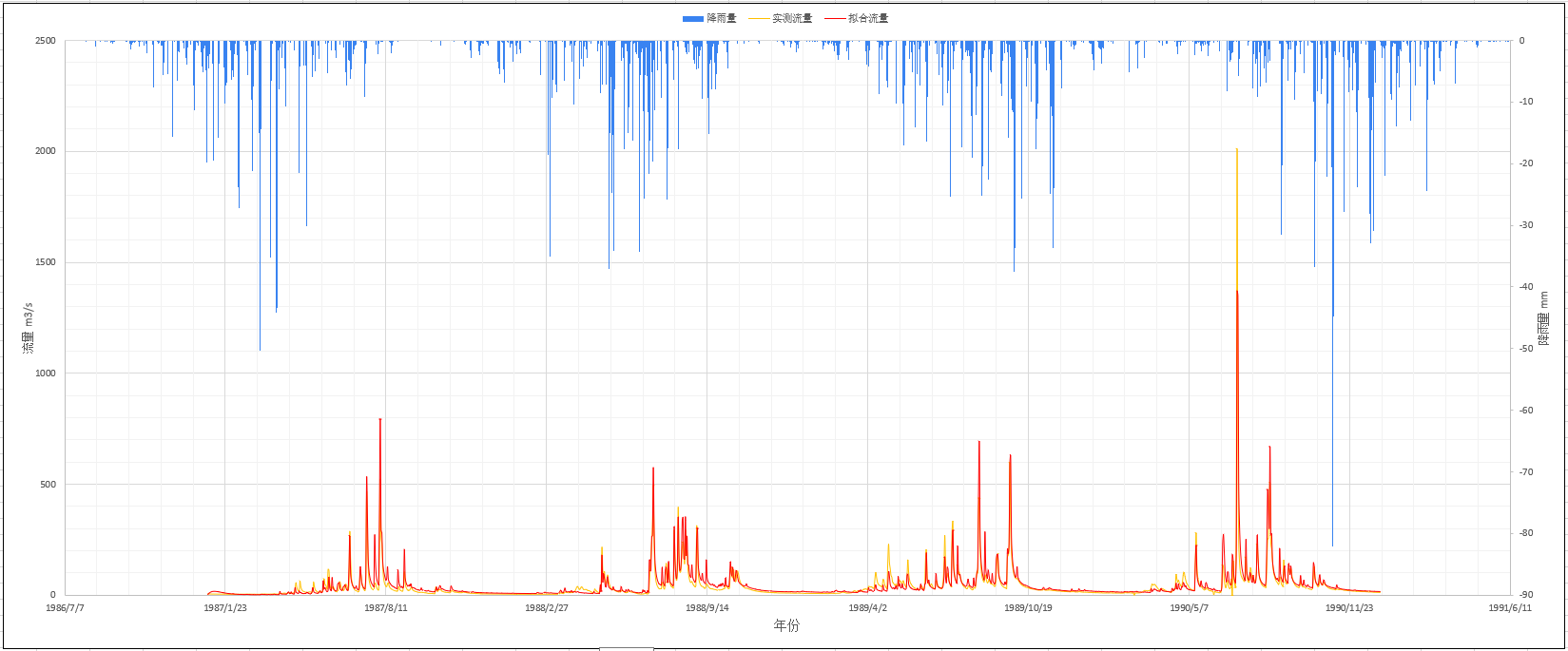


图4. 1 降雨洪水拟合过程

选取其中拟合效果较好的两场洪水（洪号分别为 870718、900516）说明拟

合效果。

如下图4.2，在870718号洪水的拟合中，拟合流量过程与实测流量过程形状大致相同，且均明显符合洪水相较于降雨过程的滞后性，但拟合洪水过程相较于实测洪水过程流量偏大。

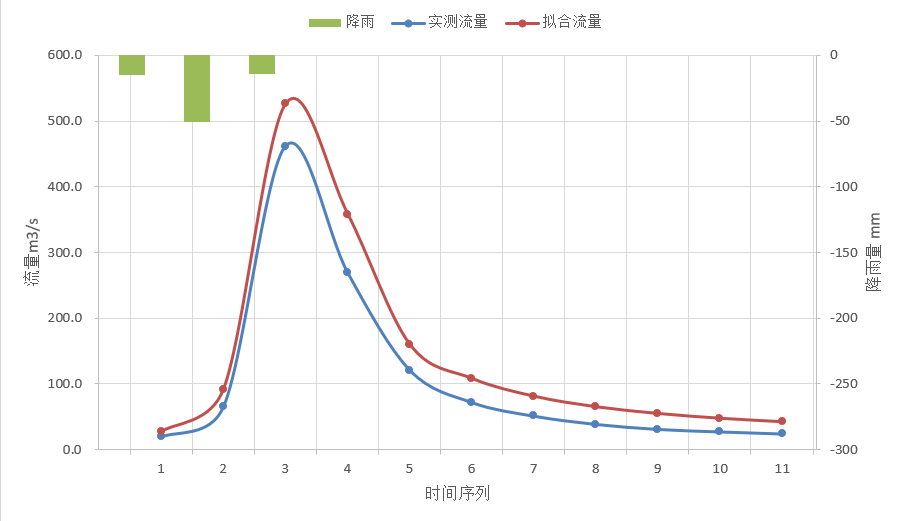


图4. 2 870718号洪水过程拟合

在900516号洪水的拟合中， 实测流量过程与拟合流量过程大致相同，但实测流量对于第二场降雨的响应不明显，而拟合流量则相对于明显一些，导致这一现象的原因可能是实测流量过程有一定的的误差，与此同时也要注意拟合流量只是一个参考过程，不能说实测流量一定是有问题的，只是有这个可能性。

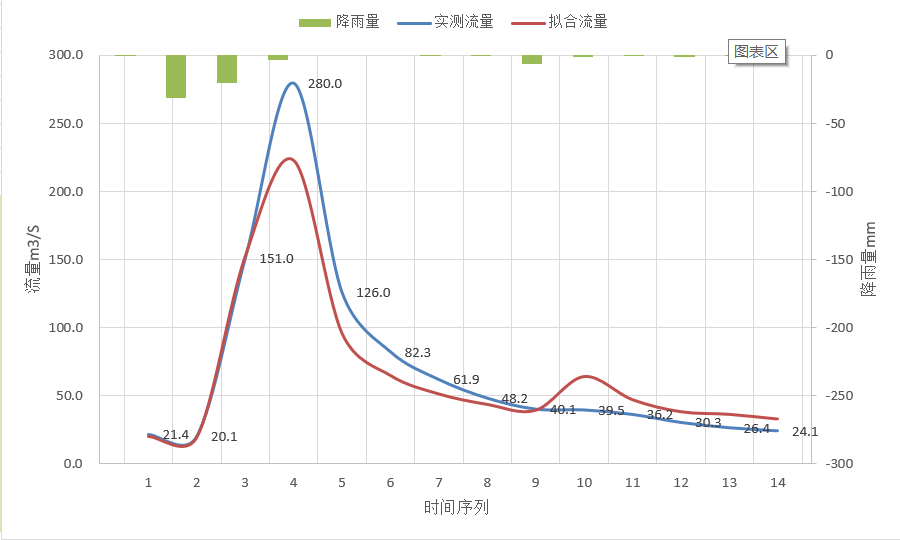


图4. 3 900516号洪水过程拟合

1. 存在的问题与改进方案

对1987~1990年的场次洪水预报模拟情况分析发现，有场洪水的模拟效果不是很好，通过对些场次洪水实测流量与模拟流量的比较发现，模拟洪峰的峰现时间与实测洪峰相比，相差在可接受的范围内，且无明显的提前或拖后。但洪峰大小比实测洪峰偏小。

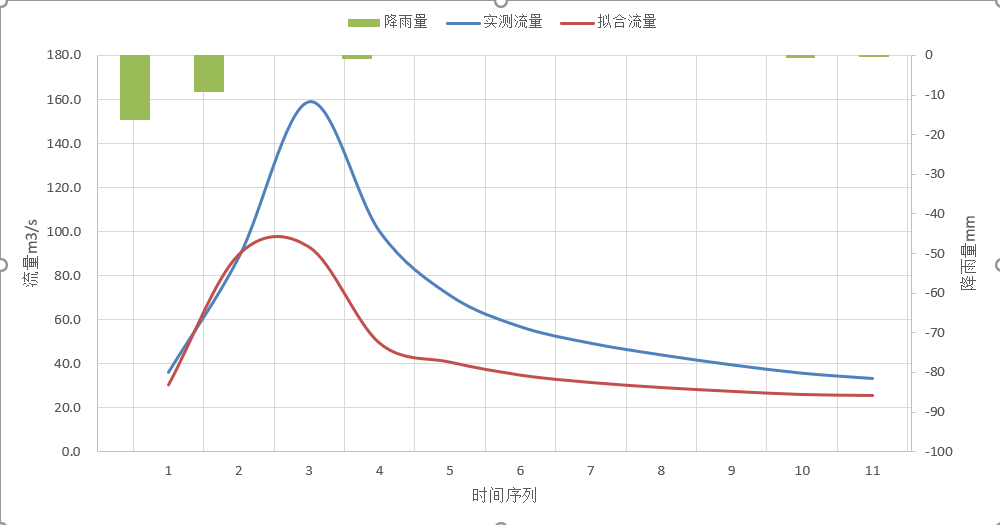


图4. 4 效果较差的拟合

综合来看模拟的径流过程趋势上与实际径流过程基本一致，但是径流量差别较大。可能有两个方面的原因：

1）对于新安江模型本身来讲，可能会把突变的地方通过均摊的方式，使其平滑化。而实际情况中，这种突变的流量情况是会存在的，而洪水过程中这种情况会更加明显。所以模拟洪峰会比实测洪峰偏小。

2）上述问题的存在可能与率定期的单位线有关，同时也可能与水文测站的数量有关。水文站的数量越多，单位线的单位时段越短，模拟径流与实测径流应该越接近。可以考虑适当将单位线进行调整，来使其精度增加。

3）率定期参数误差较大。在上述率定期的检验中，只有不到70%的合格率，因此导致率定的参数误差较大，可以考虑修改参数重新率定。

参考文献

[1] 赵人俊.流域水文模型的比较分析研究.水文，1989，（6）

[2] 赵人俊.新安江模型参数的分析.水文，1988，（6）：2-9.

[3] 谭炳卿.水文模型参数自动优选方法的比较分析.水文.1996（5）：8-13.

结语

本次课设是我做的第一个课设，事实上我比较喜欢这种需要自己不断地查资料、研究、与同学讨论去解决问题的这一过程，而且我个人热爱编程，本次课设计算程序我使用python进行编写，以前我觉得python做科学计算不如MATLAB好用，但是通过本次的课设我发现事实上由于有强大的科学计算相关的库，例如scipy，numpy，pandas，因此python做科学计算也很方便。

在课设初期我的计划是使用面向对象的方法进行编程，事实上我也确实是这么做的，而这样的问题就在于，本来并没有新安江这样一个现成的对象使用，因此需要自己从类、方法的设计开始，这也意味着我需要比使用C语言，matlab的同学使用更多的代码量，花费更多的时间，但问题的冲突就在于本次课设的时间上，课设的同时伴随两场考试，因此时间非常紧张，而这也是导致我没有深入检查上述错误的原因之一。

但无论如何，本次课设我还是有很大的收获，在学习课程时，我们只是零碎的知道某一项参数或者某一个方法如何去算，但是并不知道他有什么用，而课设是把这些东西串联到一起，就像是一条看不见的线，把学过的东西串在一起，真正达到会学会用。同时，也加深了对于新安江模型—这一我国水文事业上的一个里程碑式成果的理解。虽然这中间的过程有些曲折，从刚开始的无从下手到现在对于新安江模型计算过程的如数家珍，但是我还是只了解了新安江模型的冰山一角，例如在看到赵人俊先生的《新安江模型的参数分析》一文时还是不能理解其中的原理，但这也是我今后学习的动力。

总之，本次课设我学到了很多，虽然时间赶任务急，但是还是有一定的效果。

附件

1. 计算代码
2. **import** numpy **as** np  
   **import** pandas **as** pd  
   **import** os  
   **import** matplotlib.pyplot **as** plt  
   **from** tqdm **import \****# 产流计算  
   # 本类是产流计算类，只用于为蓄满产流类（Dunne）， 蒸法类（evaporation）初始化相关参数***class RunOffGeneration**(object)**:  
    def \_\_init\_\_**(self, *args*)**:** self.IMP **=** *args*[**'IMP'**]  
    self.WM **=** *args*[**'WM'**]  
    self.WUM **=** *args*[**'WUM'**]  
    self.WLM **=** *args*[**'WLM'**]  
    self.WDM **=** *args*[**'WDM'**]  
    self.B **=** *args*[**'B'**]  
    self.C **=** *args*[**'C'**]  
    self.FE **=** *args*[**'FE'**]  
    self.data\_original **=** *args*[**'data'**]  
    self.data **=** *args*[**'data'**]  
    self.\_init\_columns()  
     
    *# 重新初始化列名* **def \_init\_columns**(self)**:** self.data **=** self.data.reindex(columns**=**[**'date'**,  
    **'P'**, **'pe'**,  
    **'E'**, **'ep'**, **'eu'**, **'el'**, **'ed'**, **'e'**,  
    **'wu'**, **'wl'**, **'wd'**, **'w'**,  
    **'R'**,  
    **'Q'**])  
     
     
   *# 蓄满产流计算  
   # 继承自runoffgeneration类***class Dunne**(RunOffGeneration)**:  
    def \_\_init\_\_**(self, *args*)**:** super().\_\_init\_\_(*args*) *# 调用父类初始化参数* self.W **=** self.WM **\*** self.FE  
    self.WMM **=** self.\_wmm()  
     
    *# 内部计算WMM* **def \_wmm**(self)**:  
    return** self.WM **\*** (1 **+** self.B)**/**(1 **-** self.IMP)  
     
    **def \_a**(self, *i*)**:  
    return** self.WMM **\*** (1 **-** (1 **-** self.data.loc[*i*, **'w'**] **/** self.WM) **\*\*** (1 **/** (1 **+** self.B)))  
     
    **def \_pe**(self, *i*)**:** p **=** self.data.loc[*i*, **'P'**]  
    pe **=** p **\*** (1 **-** self.IMP) **-** self.data.loc[*i*, **'e'**]  
    self.data.loc[*i*, **'pe'**] **=** pe  
    **return** pe  
     
    *# 计算过程总控制器* **def calculate**(self, *i*)**:** a **=** self.WMM **\*** (1 **-** (1 **-** self.data.loc[*i*, **'w'**] **/** self.WM) **\*\*** (1 **/** (1 **+** self.B)))  
    PE **=** self.\_pe(*i*)  
    **if** PE **<** 0**:** self.data.loc[*i*, **'R'**] **=** 0  
    **else:  
    if** a **+** PE **<=** self.WMM**:** self.data.loc[*i*, **'R'**] **=** PE **+** self.data.loc[*i*, **'w'**] **-** self.WM **+** self.WM **\*** (1 **-** (PE **+** a) **/** self.WMM) **\*\*** (self.B **+** 1)  
    **else:** self.data.loc[*i*, **'R'**] **=** PE **-** (self.WM **-** self.data.loc[*i*, **'w'**])  
     
    *# 同步数据  
    # 由于将蒸法与产流分开两个类，因此二者之间的数据传递之前需要同步* **def sync\_data**(self, *df*)**:** self.data.update(*df*)  
     
     
   *# 蒸发计算类***class Evaporation**(RunOffGeneration)**:  
    def \_\_init\_\_**(self, *params*)**:** *# 调用父类初始化参数* super().\_\_init\_\_(*params*)  
    self.kc **=** *params*[**'KC'**]  
    *# print("debug> kc == ", self.kc)* self.row **=** len(self.data.index)  
    self.column **=** 10  
    self.\_init\_columns\_()  
     
    *# 初始化列名* **def \_init\_columns\_**(self)**:** self.data.loc[0, **'wu'**] **=** self.FE **\*** self.WUM  
    self.data.loc[0, **'wl'**] **=** self.FE **\*** self.WLM  
    self.data.loc[0, **'wd'**] **=** self.FE **\*** self.WDM  
    self.data.loc[0, **'w'**] **=** self.data.loc[0, **'wu'**] **+** self.data.loc[0, **'wl'**] **+** self.data.loc[0, **'wd'**]  
    self.data.loc[0, **'eu'**] **=** 0  
    self.data.loc[0, **'el'**] **=** 0  
    self.data.loc[0, **'ed'**] **=** 0  
    self.data.loc[0, **'e'**] **=** 0  
    self.data.loc[**:**, **'ep'**] **=** self.ep()  
     
    *# 计算Ep* **def ep**(self)**:** ep **=** [x **\*** self.kc **for** x **in** self.data.loc[**:**, **'E'**]]  
    **return** ep  
     
    *# 计算EU， EL， ED， E* **def eu\_el\_ed**(self, *i*)**:** wu, p, ep, wl, wd **=** self.data.loc[*i*, [**'wu'**, **'P'**, **'ep'**, **'wl'**, **'wd'**]]  
    p **=** p **\*** (1 **-** self.IMP) *# 考虑不透水区域，其所占比例为IMP  
    # 常规计算的使用分支的方法  
    # if wu + p >= ep:  
    # self.data.loc[i, 'eu'] = ep  
    # self.data.loc[i, 'el'] = 0  
    # self.data.loc[i, 'ed'] = 0  
    # else:  
    # eu = wu + p  
    # self.data.loc[i, 'eu'] = eu  
    # if wl >= self.C \* self.WLM:  
    # self.data.loc[i, 'el'] = ((ep - eu) \* wl) / self.WLM  
    # self.data.loc[i, 'ed'] = 0  
    # else:  
    # if wl >= self.C \* (ep - eu):  
    # self.data.loc[i, 'el'] = self.C \* (ep - eu)  
    # self.data.loc[i, 'ed'] = 0  
    # else:  
    # el = wl  
    # self.data.loc[i, 'el'] = el  
    # self.data.loc[i, 'ed'] = self.C \* (ep - eu) - el  
    # if self.data.loc[i, 'ed'] > wd:  
    # self.data.loc[i, 'ed'] = wd  
    # 简化计算分支的方法  
    # if wu + p < ep:  
    # eu = wu + p  
    # self.data.loc[i, 'eu'] = eu  
    # if wl < self.C \* self.WLM:  
    # if self.C \* (ep - eu) > wl:  
    # el = wl  
    # self.data.loc[i, 'el'] = el  
    # if self.data.loc[i, 'ed'] < wd:  
    # self.data.loc[i, 'ed'] = self.C \* (ep - eu) - el  
    # self.data.loc[i, 'ed'] = wd  
    # self.data.loc[i, 'el'] = self.C \* (ep - eu)  
    # self.data.loc[i, 'ed'] = 0  
    # self.data.loc[i, 'el'] = ((ep - eu) \* wl) / self.WLM  
    # self.data.loc[i, 'ed'] = 0  
    # self.data.loc[i, 'eu'] = ep  
    # self.data.loc[i, 'el'] = 0  
    # self.data.loc[i, 'ed'] = 0* self.data.loc[*i*, **'e'**] **=** self.data.loc[*i*, **'eu'**] **+** self.data.loc[*i*, **'el'**] **+** self.data.loc[*i*, **'ed'**] *# 总E  
     
    # 计算WU， WL， WD， W* **def wu\_wl\_wd**(self, *j*)**:** wu, wl, wd, p, eu, el, ed, R **=** self.data.loc[*j* **-** 1, [**'wu'**, **'wl'**, **'wd'**, **'P'**, **'eu'**, **'el'**, **'ed'**, **'R'**]]  
    wu **=** wu **+** p **-** self.data.loc[*j* **-** 1, **'eu'**] **-** R  
    *# 同上，本处使用简化计算分支方法，减少代码缩进* **if** wu **<** self.WUM**:** self.data.loc[*j*, **'wu'**] **=** wu  
    self.data.loc[*j*, **'wl'**] **=** wl **-** el  
    self.data.loc[*j*, **'wd'**] **=** wd **-** ed  
    self.data.loc[*j*, **'w'**] **=** self.data.loc[*j*, **'wu'**] **+** self.data.loc[*j*, **'wl'**] **+** self.data.loc[*j*, **'wd'**]  
    **return** 1  
    self.data.loc[*j*, **'wu'**] **=** self.WUM  
    wl **=** wl **-** el **+** wu **-** self.WUM  
    **if** wl **<** self.WLM**:** self.data.loc[*j*, **'wl'**] **=** wl  
    self.data.loc[*j*, **'wd'**] **=** wd **-** ed  
    self.data.loc[*j*, **'w'**] **=** self.data.loc[*j*, **'wu'**] **+** self.data.loc[*j*, **'wl'**] **+** self.data.loc[*j*, **'wd'**]  
    **return** 2  
    self.data.loc[*j*, **'wl'**] **=** self.WLM  
    wd **=** wd **-** ed **+** wl **-** self.WLM  
    **if** wd **<** self.WDM**:** self.data.loc[*j*, **'wd'**] **=** wd  
    self.data.loc[*j*, **'w'**] **=** self.data.loc[*j*, **'wu'**] **+** self.data.loc[*j*, **'wl'**] **+** self.data.loc[*j*, **'wd'**]  
    **return** 3  
    self.data.loc[*j*, **'wd'**] **=** self.WDM  
    self.data.loc[*j*, **'w'**] **=** self.data.loc[*j*, **'wu'**] **+** self.data.loc[*j*, **'wl'**] **+** self.data.loc[*j*, **'wd'**]  
     
    *# 产流计算中间件* **def dunne\_middle\_ware**(self, *i*, *dunne*)**:** *dunne*.calculate(*i*)  
     
    *# 同步数据* **def sync\_data**(self, *df*)**:** self.data.update(*df*)  
     
    *# 计算总控制器* **def calculate**(self, *i*, *dunne*)**:** self.eu\_el\_ed(*i*)  
    *dunne*.sync\_data(self.data)  
    self.dunne\_middle\_ware(*i*, *dunne*)  
    self.sync\_data(*dunne*.data)  
    self.wu\_wl\_wd(*i* **+** 1)  
     
     
   *# 产流计算总控制器***class RunOffCalculator**(object)**:  
    def \_\_init\_\_**(self, *params*)**:** self.evaporation **=** Evaporation(*params*)  
    self.dunne **=** Dunne(*params*)  
    self.row **=** len(*params*[**'data'**].index)  
     
    **def calculate**(self)**:  
    for** i **in** tqdm(range(self.row))**:** self.evaporation.calculate(i, self.dunne)  
     
     
   *# 水源划分类***class WaterSourceSplit**(object)**:  
    def \_\_init\_\_**(self, *params*)**:** self.WM **=** *params*[**'WM'**]  
    self.FE **=** *params*[**'FE'**]  
    self.B **=** *params*[**'B'**]  
    self.SM **=** *params*[**'SM'**]  
    self.EX **=** *params*[**'EX'**]  
    self.KG **=** *params*[**'KG'**]  
    self.KKG **=** *params*[**'KKG'**]  
    self.KSS **=** *params*[**'KSS'**]  
    self.KKSS **=** *params*[**'KKSS'**]  
    self.data **=** *params*[**'data'**]  
    self.\_row **=** len(self.data.index)  
    self.\_init\_columns()  
     
    *# 初始化列名* **def \_init\_columns**(self)**:** self.data **=** self.data.reindex(columns**=**[**'date'**,  
    **'P'**, **'pe'**,  
    **'E'**, **'ep'**, **'eu'**, **'el'**, **'ed'**, **'e'**,  
    **'wu'**, **'wl'**, **'wd'**, **'w'**,  
    **'FR'**, **'S'**,  
    **'R'**, **'RS'**, **'RI'**, **'RG'**,  
    **'Q'**])  
    self.data.loc[0, **'S'**] **=** self.SM **\*** self.FE  
     
    *# 水源划分* **def runoff**(self, *i*)**:** *# s0 = self.data.loc[0, 'S']* MS **=** self.SM **\*** (1 **+** self.EX)  
    pe, w, s, r **=** self.data.loc[*i*, [**'pe'**, **'w'**, **'S'**, **'R'**]]  
    *# s0, fr0 = self.data.loc[i-1, ['S', 'FR']]* **if** pe **<=** 0**:** fr **=** 1 **-** (1 **-** w **/** self.WM)**\*\***(self.B **/** (1 **+** self.B))  
    self.data.loc[*i*, **'FR'**] **=** fr  
    *# s = s0 \* fr0 /* self.data.loc[*i*, **'RS'**] **=** 0  
    self.data.loc[*i*, **'RI'**] **=** s **\*** self.KSS **\*** fr  
    self.data.loc[*i*, **'RG'**] **=** s **\*** self.KG **\*** fr  
    self.data.loc[*i* **+** 1, **'S'**] **=** (1 **-** self.KSS **-** self.KG) **\*** s  
    **else:** fr **=** r **/** pe  
    self.data.loc[*i*, **'FR'**] **=** fr  
    *# AU = MS \* (1 - (1 - (s0\*(fr0/fr)) / self.SM)\*\*(1 / (1 + self.EX)))* AU **=** MS **\*** (1 **-** (1 **-** s **/** self.SM)**\*\***(1 **/** (1 **+** self.EX)))  
    **if** pe **+** AU **<** MS**:** *# self.data.loc[i, 'RS'] = fr \* (pe + s0 \* fr0 / fr - self.SM + self.SM \* (1 - (pe / AU) / MS) \*\* (  
    # self.EX + 1))* tmp **=** self.SM **\*** (1 **-** (pe **+** AU) **/** MS) **\*\*** (self.EX **+** 1)  
    self.data.loc[*i*, **'RS'**] **=** fr **\*** (pe **+** s **-** self.SM **+** tmp)  
    self.data.loc[*i*, **'RI'**] **=** fr **\*** self.KSS **\*** (self.SM **-** tmp)  
    self.data.loc[*i*, **'RG'**] **=** fr **\*** self.KG **\*** (self.SM **-** tmp)  
    self.data.loc[*i* **+** 1, **'S'**] **=** (1 **-** self.KSS **-** self.KG) **\*** (self.SM **-** tmp)  
    **else:** self.data.loc[*i*, **'RS'**] **=** fr **\*** (pe **-** self.SM **+** s)  
    self.data.loc[*i*, **'RI'**] **=** fr **\*** self.SM **\*** self.KSS  
    self.data.loc[*i*, **'RG'**] **=** fr **\*** self.KG **\*** self.SM  
    self.data.loc[*i* **+** 1, **'S'**] **=** self.SM **\*** (1 **-** self.KSS **\*** self.KG)  
     
    **def calculate**(self)**:  
    for** i **in** tqdm(range(self.\_row))**:** self.runoff(i)  
     
     
   *# 汇流计算类***class Concentrate**(object)**:  
    def \_\_init\_\_**(self, *params*)**:** self.IMP **=** *params*[**'IMP'**]  
    self.UH **=** *params*[**'UH'**]  
    self.KKSS **=** *params*[**'KKSS'**]  
    self.KKG **=** *params*[**'KKG'**]  
    self.A **=** *params*[**'A'**]  
    self.uh\_len **=** len(self.UH)  
    self.data **=** *params*[**'data'**]  
    self.delta\_t **=** 24  
    self.row **=** len(self.data.index)  
    self.\_init\_columns()  
     
    **def \_init\_columns**(self)**:** self.data **=** self.data.reindex(columns**=**[**'date'**,  
    **'P'**, **'pe'**,  
    **'ep'**, **'eu'**, **'el'**, **'ed'**, **'e'**,  
    **'wu'**, **'wl'**, **'wd'**, **'w'**,  
    **'FR'**, **'S'**,  
    **'R'**, **'RS'**, **'RI'**, **'RG'**,  
    **'QRS'**, **'QRI'**, **'QRG'**, **'QRT'**, **'Q'**])  
    self.data.loc[0, [**'QRI'**, **'QRG'**]] **=** 0.5, 0.0  
    self.data.loc[**:**, **'QRS'**] **=** 0  
     
    *# 主计算方法* **def calculate**(self)**:  
    for** i **in** tqdm(range(self.row))**:  
    for** j **in** range(self.uh\_len)**:** rs **=** self.data.loc[i, **'RS'**]  
    r\_imp **=** self.data.loc[i, **'P'**] **\*** self.IMP  
    uh **=** self.UH[j]**/**10  
    **try:** self.data.loc[i**+**j, **'QRS'**] **+=** (rs **+** r\_imp) **\*** uh  
    **except** Exception **as** e**:** print(**'[WARNING]:'**, e)  
    QRI **=** self.data.loc[i, **'QRI'**] *# 壤中流* QRG **=** self.data.loc[i, **'QRG'**] *# 地下径流* RI **=** self.data.loc[i, **'RI'**] *# 壤中流产流* RG **=** self.data.loc[i, **'RG'**] *# 地下径流产流* self.data.loc[i, **'QRT'**] **=** self.data.loc[i, **'QRS'**] **+** self.data.loc[i, **'QRI'**] **+** self.data.loc[i, **'QRG'**] *# 总计算径流* self.data.loc[i**+**1, **'QRI'**] **=** QRI **\*** self.KKSS **+** RI **\*** (1 **-** self.KKSS) **\*** self.A **/** (3.6 **\*** self.delta\_t) *# 计算下一个QRI* self.data.loc[i**+**1, **'QRG'**] **=** QRG **\*** self.KKG **+** RG **\*** (1 **-** self.KKG) **\*** self.A **/** (3.6 **\*** self.delta\_t)  
     
     
   *# 洪水场次分割***class FloodSplit**(object)**:  
    def \_\_init\_\_**(self, *params*)**:** self.flood\_data **=** *params*[**'flood\_data'**]  
    self.excel\_data **=** *params*[**'excel\_data'**]  
    self.row\_flood\_data **=** len(self.flood\_data.index)  
    self.flood\_start\_index **=** []  
    self.flood\_stop\_index **=** []  
     
    *# 主分割方法，使用索引号定位* **def split\_flood\_data**(self)**:** *# 1. 分割洪号、起止时间  
    # 起 = '19' + 洪号前两个 + '-' + 起时间前两个 + '-' + 起时间后两个  
    # 止 = '19' + 洪号前两个 + '-' + 止时间前两个 + '-' + 止时间后两个* flood\_num **=** np.array(self.flood\_data.loc[**:**, **'flood\_num'**])  
    flood\_year **=** []  
    flood\_start\_stop **=** np.array(self.flood\_data.loc[**:**, **'start\_stop'**])  
    flood\_one\_start **=** []  
    flood\_one\_stop **=** []  
    flood\_start **=** []  
    flood\_stop **=** []  
    **for** flood **in** flood\_num**:** flood\_year.append(**'19' +** str(flood)[0**:**2])  
    **for** flood **in** flood\_start\_stop**:** flood\_one\_start.append(str(flood)[0**:**2] **+ '-' +** str(flood)[3**:**5])  
    flood\_one\_stop.append(str(flood)[6**:**8] **+ '-' +** str(flood)[9**:**11])  
    **for** year, start, stop **in** zip(flood\_year, flood\_one\_start, flood\_one\_stop)**:** flood\_start.append(year **+ '-' +** start)  
    flood\_stop.append(year **+ '-' +** stop)  
    print(year **+ '-' +** start **+ '-->' +** year **+ '-' +** stop)  
    *# 2. 定位起止时间索引号* excel\_data\_date **=** self.excel\_data.loc[**:**, **'date'**]  
    excel\_data\_date **=** [str(x.date()) **for** x **in** excel\_data\_date]  
    **for** start, stop **in** zip(flood\_start, flood\_stop)**:** self.flood\_start\_index.append(excel\_data\_date.index(start))  
    self.flood\_stop\_index.append(excel\_data\_date.index(stop))  
     
    *# 计算洪水总量* **def sum\_flood**(self)**:  
    for** start, stop, row **in** zip(self.flood\_start\_index, self.flood\_stop\_index, range(self.row\_flood\_data))**:** flood\_measured **=** np.array(self.excel\_data.loc[start**:**stop, **'Q'**])  
    flood\_simulated **=** np.array(self.excel\_data.loc[start**:**stop, **'QRT'**])  
    flood\_p **=** np.array(self.excel\_data.loc[start**:**stop, **'P'**])  
    *# 确定性系数计算* ave **=** flood\_measured.sum() **/** len(flood\_measured) *# 实测均值* up **=** np.array([(yc **-** y0) **\*\*** 2 **for** yc, y0 **in** zip(flood\_simulated, flood\_measured)]).sum() *# 分子* down **=** np.array([(y0 **-** ave) **\*\*** 2 **for** y0 **in** flood\_measured]).sum() *# 分母* R\_2 **=** 1 **-** up **/** down *# 确定性系数* self.flood\_data.loc[row, **'R^2'**] **=** R\_2 *# 赋值给相应dataframe  
    # 洪水总量计算* self.flood\_data.loc[row, **'sum\_measured\_flood'**] **=** flood\_measured.sum()  
    self.flood\_data.loc[row, **'sum\_simulated\_flood'**] **=** flood\_simulated.sum()  
    self.flood\_data.loc[row, **'P'**] **=** flood\_p.sum()  
     
    *# 洪峰计算* **def peak**(self)**:** peak\_measured **=** []  
    peak\_simulated **=** []  
    **for** start, stop, row **in** zip(self.flood\_start\_index, self.flood\_stop\_index, range(self.row\_flood\_data))**:** *# 实测洪峰* flood\_measured **=** self.excel\_data.loc[start**:**stop, [**'date'**, **'Q'**]]  
    flood\_measured.index **=** range(len(flood\_measured))  
    max\_ **=** flood\_measured.loc[**:**, **'Q'**].max()  
    self.flood\_data.loc[row, **'measured\_peak'**] **=** max\_  
    index **=** list(flood\_measured.loc[**:**, **'Q'**]).index(max\_) *# 洪峰索引* date **=** flood\_measured.loc[index, **'date'**] *# 峰现时间* self.flood\_data.loc[row, **'measured\_peak\_time'**] **=** date  
    peak\_measured.append((max\_, date))  
    peak\_measured.append((flood\_measured.max(), ))  
    *# 计算洪峰* flood\_simulated **=** self.excel\_data.loc[start**:**stop, [**'date'**, **'QRT'**]]  
    flood\_simulated.index **=** range(len(flood\_simulated))  
    max\_ **=** flood\_simulated.loc[**:**, **'QRT'**].max()  
    self.flood\_data.loc[row, **'simulated\_peak'**] **=** max\_  
    index **=** list(flood\_simulated.loc[**:**, **'QRT'**]).index(max\_)  
    date **=** flood\_simulated.loc[index, **'date'**]  
    self.flood\_data.loc[row, **'simulated\_peak\_time'**] **=** date  
    peak\_simulated.append((max\_, date))  
    self.\_\_setattr\_\_(**'peak\_measured'**, peak\_measured)  
    self.\_\_setattr\_\_(**'peak\_simulated'**, peak\_simulated)  
    **return** peak\_measured, peak\_simulated  
     
    *# 主计算控制器* **def calculate**(self)**:** self.split\_flood\_data()  
    self.sum\_flood()  
    self.peak()  
     
     
   *# 分析类***class Analyst**(object)**:  
    def \_\_init\_\_**(self, *params*)**:** self.A **=** *params*[**'A'**]  
    self.pr\_data **=** *params*[**'pr\_data'**]  
    self.flood\_data\_all **=** *params*[**'flood\_data\_all'**]  
    self.flood\_data **=** *params*[**'flood\_data'**]  
    self.\_init\_columns()  
     
    **def \_init\_columns**(self)**:** self.flood\_data\_all.reindex(columns**=**[**'NO'**,  
    **'flood\_num'**,  
    **'P'**,  
    **'RO'**,  
    **'RC'**,  
    **'abs\_error'**,  
    **'relative\_error'**,  
    **'R^2'**])  
     
    *# 误差计算  
    # 不同于上述使用定位索引号的方法进行索引  
    # 本处使用直接更改索引号为洪号，再使用洪号直接定位数据* **def error**(self)**:** flood\_num **=** self.flood\_data.loc[**:**, **'flood\_num'**]  
    self.flood\_data\_all.index **=** list(self.flood\_data\_all.loc[**:**, **'flood\_num'**])  
    self.flood\_data.index **=** flood\_num  
    ro\_ **=** self.flood\_data.loc[**:**, **'sum\_measured\_flood'**]  
    rc\_ **=** self.flood\_data.loc[**:**, **'sum\_simulated\_flood'**]  
    ro\_ **=** [r **\*** 3600 **\*** 24 **/** (self.A **\*** 10 **\*\*** 3) **for** r **in** ro\_]  
    rc\_ **=** [r **\*** 3600 **\*** 24 **/** (self.A **\*** 10 **\*\*** 3) **for** r **in** rc\_]  
    **for** flood, ro, rc **in** zip(flood\_num, ro\_, rc\_)**:** p, r\_2 **=** self.flood\_data.loc[flood, [**'P'**, **'R^2'**]]  
    self.flood\_data\_all.loc[flood, **'abs\_error'**] **=** rc **-** ro *# 绝对误差* self.flood\_data\_all.loc[flood, **'P'**] **=** p *# 场次总净雨* self.flood\_data\_all.loc[flood, **'RO'**] **=** ro *# 场次实测总径流* self.flood\_data\_all.loc[flood, **'RC'**] **=** rc *# 场次计算总径流* self.flood\_data\_all.loc[flood, **'R^2'**] **=** r\_2 *# 确定性系数* self.flood\_data\_all.loc[flood, **'relative\_error'**] **=** 100 **\*** (rc **-** ro) **/** ro *# 相对误差* self.flood\_data\_all.index **=** range(len(self.flood\_data\_all.index)) *# 恢复index索引* self.flood\_data.index **=** range(len(self.flood\_data.index)) *# 恢复index索引  
     
    # 分析控制器* **def analyse**(self)**:** self.error()  
     
     
   *# 主函数***if** \_\_name\_\_ **== '\_\_main\_\_':** *# 文件路径处理* WORKSPACE **=** os.path.abspath(**'..'**)  
    path\_docs **=** os.path.join(WORKSPACE, **'docs'**) *# 数据路径* path\_save\_to **=** os.path.join(path\_docs, **'baoRiverDataCal.xlsx'**) *# 存储路径* path\_source\_data **=** os.path.join(path\_docs, **'baoRiverData.xlsx'**)  
    excel\_data **=** pd.read\_excel(path\_source\_data, sheet\_name**='Sheet1'**)  
    flood\_data **=** pd.read\_excel(path\_source\_data, sheet\_name**='Sheet2'**)  
    flood\_data\_all **=** pd.read\_excel(path\_source\_data, sheet\_name**='Sheet3'**) *# 读取原始数据到dataframe中* plt.plot(excel\_data.loc[**:**, **'date'**], excel\_data.loc[**:**, **'Q'**])  
    *# 蒸发相关  
    # 1.计算蒸发， 采用三层蒸发  
    # 0.常量定义* WM, WUM, WLM, WDM, C, B, FE, KC, IMP **=** 115, 20, 25, 70, 0.16, 1.75, 0.8, 1.04, 0.054  
    *# 参数定义，用字典传递* ro\_params **=** {  
    **'IMP':** IMP,  
    **'KC':** KC,  
    **'WM':** WM,  
    **'WUM':** WUM,  
    **'WLM':** WLM,  
    **'WDM':** WDM,  
    **'B':** B,  
    **'C':** C,  
    **'FE':** FE,  
    **'data':** excel\_data,  
    }  
    print(**'<debug> ###蒸发计算开始###**\n**'**)  
    ro\_calculator **=** RunOffCalculator(ro\_params) *# 实例化产流计算对象* ro\_calculator.calculate() *# 调用上述对象的主计算控制器进行计算* print(**'<debug> ###蒸发计算结束###**\n**'**)  
    *# 2.水源划分  
    # 常量定义  
    # SM --> 流域平均自由水蓄水容量  
    # EX --> 自由水蓄量分布曲线指数  
    # KG --> 自由水对地下水的日出流系数  
    # KKG --> 地下水消退系数  
    # KSS --> 自由水对壤中流的日出流系数  
    # KKSS --> 壤中流消退系数* SM, EX, KG, KKG, KSS, KKSS **=** 40, 1.9, 0.06, 0.995, 0.05, 0.885  
    ws\_params **=** {  
    **'WM':** WM,  
    **'FE':** FE,  
    **'B':** B,  
    **'SM':** SM,  
    **'EX':** EX,  
    **'KG':** KG,  
    **'KKG':** KKG,  
    **'KSS':** KSS,  
    **'KKSS':** KKSS,  
    **'data':** ro\_calculator.evaporation.data,  
    }  
    print(**'<debug> ###水源划分计算开始###**\n**'**)  
    ws\_calculator **=** WaterSourceSplit(ws\_params) *# 实例化水源划分对象* ws\_calculator.calculate()  
    print(**'<debug> ###水源划分计算结束###**\n**'**)  
    *# 水源划分结束  
    # 3.汇流开始  
    # 定义常量* A **=** 3415  
    UH **=** [0, 238.6, 63.8, 34.5, 20.6, 12.9, 8.3, 5.5, 3.6,  
    2.4, 1.6, 1.1, 0.8, 0.5, 0.4, 0.24, 0.16, 0.12,  
    0.08, 0.04, 0]  
    UH **=** np.array(UH)  
    ct\_params **=** {  
    **'IMP':** IMP,  
    **'UH':** UH,  
    **'KKG':** KKG,  
    **'A':** A,  
    **'KKSS':** KKSS,  
    **'data':** ws\_calculator.data,  
     
    }  
    print(**'<debug> ###汇流计算开始###**\n**'**)  
    *# 4.汇流计算  
    # 定义常量* ct\_calculator **=** Concentrate(ct\_params) *# 实例化汇流计算对象* ct\_calculator.calculate()  
    print(**'<debug> ###汇流计算结束###**\n**'**)  
    print(**'<debug> ###洪水分割计算开始###**\n**'**)  
    fs\_params **=** {  
    **'excel\_data':** ct\_calculator.data,  
    **'flood\_data':** flood\_data,  
    }  
    fs\_calculator **=** FloodSplit(fs\_params) *# 实例化洪水分割对象* fs\_calculator.calculate()  
    print(**'<debug> ###洪水分割计算结束###**\n**'**)  
    print(**'<debug> ###分析数据开始###'**)  
    ana\_params **=** {  
    **'A':** A,  
    **'pr\_data':** ct\_calculator.data,  
    **'flood\_data\_all':** flood\_data\_all,  
    **'flood\_data':** flood\_data,  
    }  
    analyst **=** Analyst(ana\_params) *# 实例化分析对象* analyst.analyse()  
    print(**'<debug> ###分析数据结束###'**)  
    print(**'<debug> ###绘图开始###'**)  
    plt.plot(ct\_calculator.data.loc[**:**, **'date'**], ct\_calculator.data.loc[**:**, **'QRT'**])  
    plt.legend()  
    plt.show()  
    print(**'<debug> ###绘图结束###'**)  
    print(**'<debug> ###导出数据开始###'**)  
    writer **=** pd.ExcelWriter(path\_save\_to)  
    **try:** ct\_calculator.data.to\_excel(writer, sheet\_name**='蒸发-产流-汇流计算表'**)  
    fs\_calculator.flood\_data.to\_excel(writer, sheet\_name**='次洪统计表'**)  
    analyst.flood\_data\_all.to\_excel(writer, sheet\_name**='次洪总表与分析'**)  
    **except** Exception **as** e**:** print(e)  
    writer.save()  
    writer.close()  
    print(**'<debug> ###导出数据结束###'**)