水文风险模拟分析子系统
验收汇报

总体结构

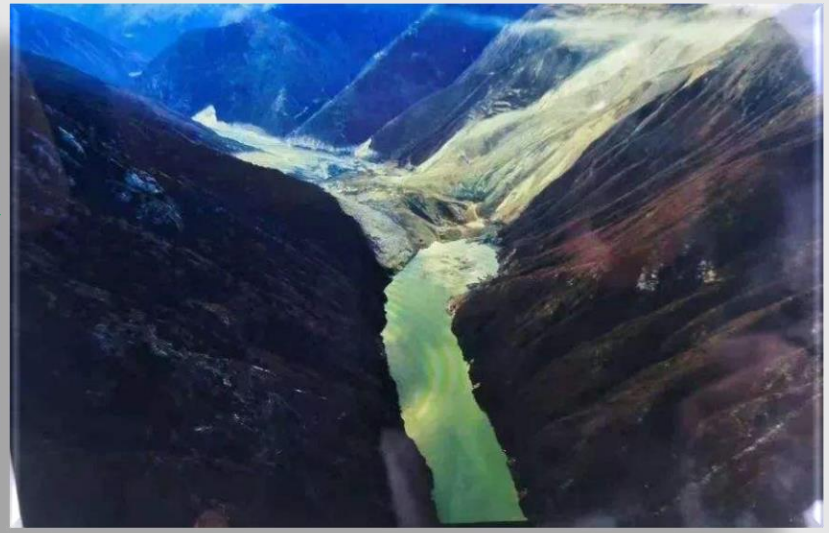
水文风险子系统模拟分析子系统

总体结构

以堰塞湖为代表的水文风险模拟分析

子流域划分及汇流单位线提取模块

1、堰塞湖来水？



一维河道淹没快速分析模块

二维区域淹没快速分析模块

4、堰塞湖风险？

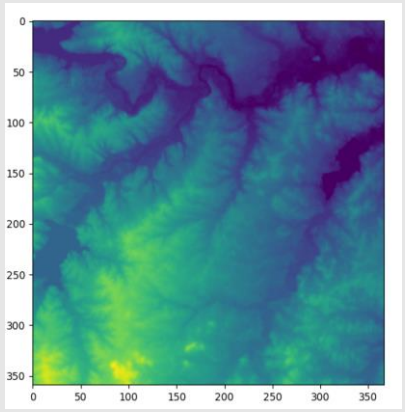
堰塞湖库容曲线提取模块

2、堰塞湖容水？

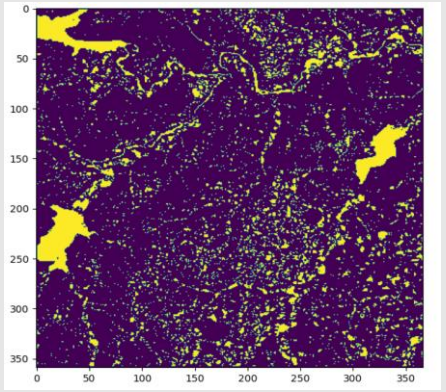
溃坝洪水分析模块

3、堰塞湖出水？

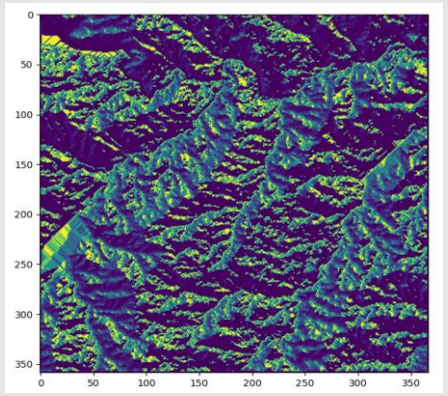
子流域划分及汇流单位线提取模块--算法流程



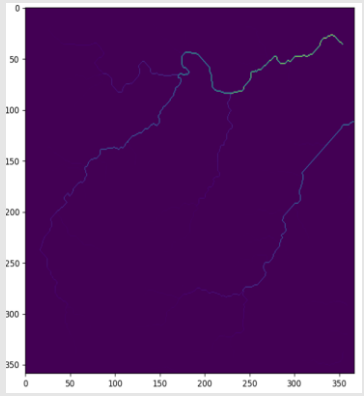
1、DEM读取



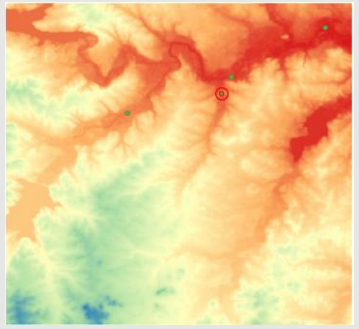
2、填洼后平面检测



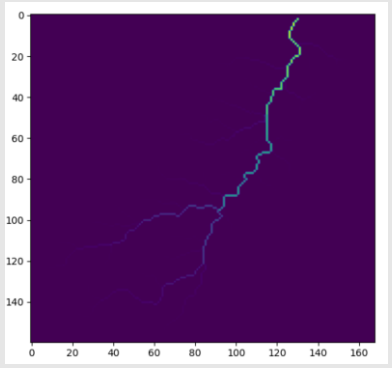
3、根据修正后的DEM计算流向



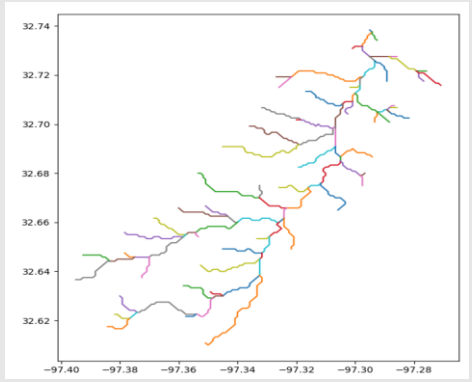
4、根据计算的流向计算流量累积



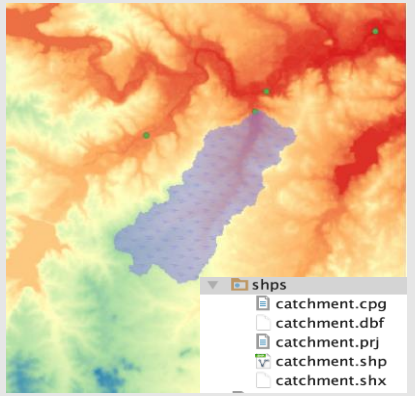
5、指定出口断面位置，确定流域范围



6、计算子流域范围内的流量累积



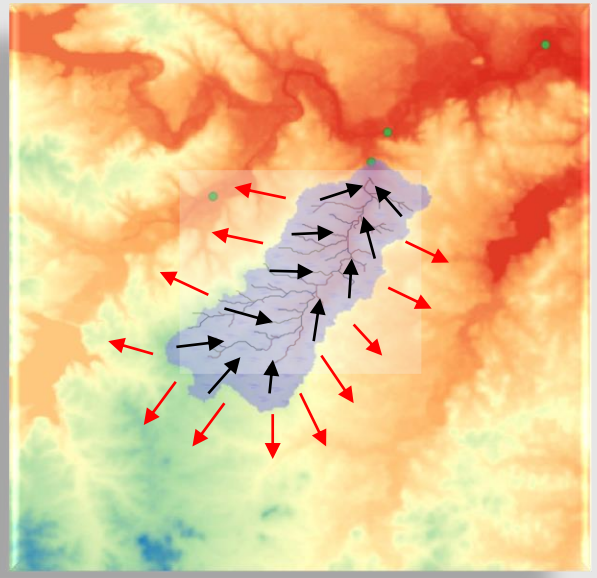
7、提取河网



8、集水流域的矢量图层输出

子流域划分及汇流单位线提取模块--关键步骤

子流域提取

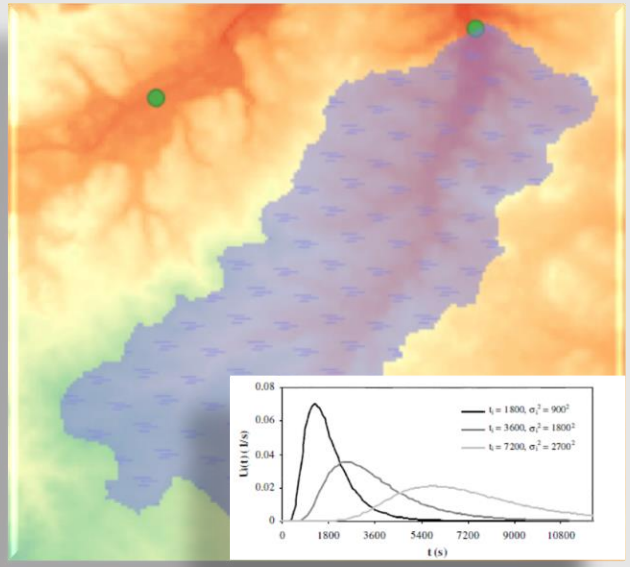


对堰塞湖上游地区进行网格剖分，计算每个网格的水流流向，划分分水岭，分水岭闭合区域即为堰塞湖的汇流子流域。

降水

水文分水岭

汇流单位线提取



基于子流域划分成果，在每个网格的流速以及汇流至子流域出口即堰塞湖的时间，关于时间累加不同时刻到达的网格水量，即为子流域的汇流单位线，也就可以预测未来堰塞湖的来水过程。

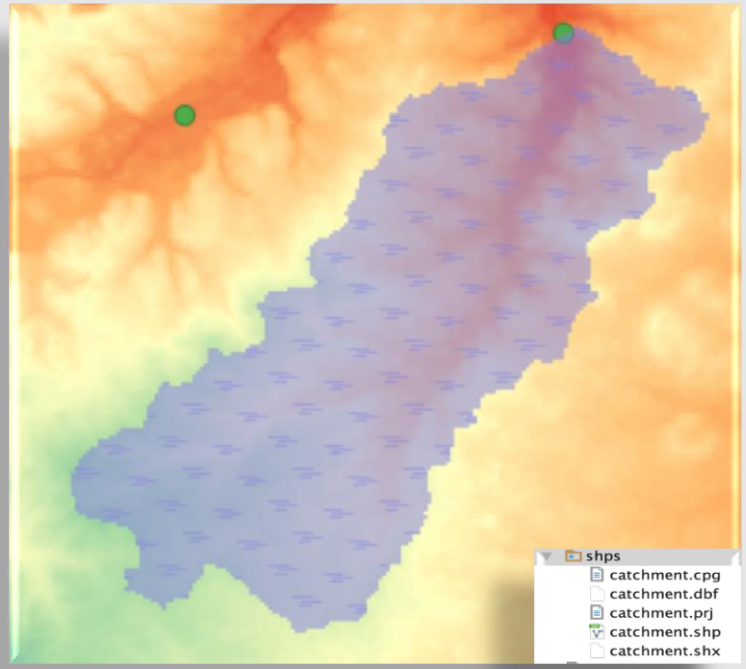
子流域划分及汇流单位线提取模块--关键代码及输出

关键代码

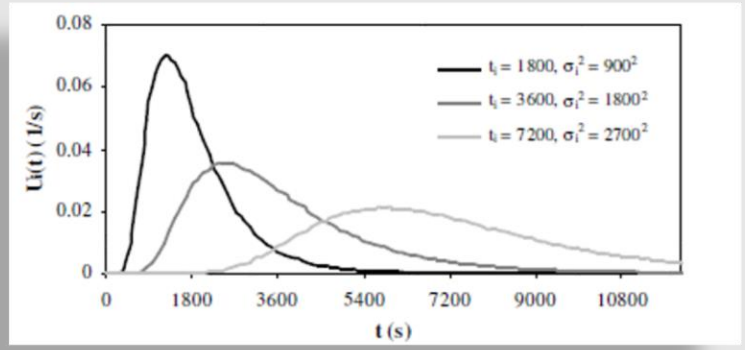
```
'''
根据出口点位置绘制子流域
---------------
初始化参数
recursionli
data:数据流方
out_name:包含
---------------
'''

def DelCatc
    self.gr

    self.gr

'''
提取河网信息
---------------
初始化参数
threshold:河
data_name:包
out_name:积
---------------
'''

def extract
    self.br
    return

'''
获取流量距离阵
---------------
初始化参数
data:包含新集
out_name:包含
---------------
'''

def getFlow
    self.gr
    return
```

```
'''
计算给定子流域对应的每个单元格的流速
---------------
初始化参数
n:曼宁系数
s:单元格坡度
return:求得每个单元格的流速
---------------
'''

def CalcuCellVelocity(self, n=None):
    if n is None:
        n = 0.025
    cell_hradius = self.CalcuHydraulicRadius()
    cell_slopes = self.getCellSlope('flow_dir', 'inflated_dem')
    # print(cell_slopes)
    if cell_hradius is None or cell_slopes is None:
        logging.info('CellHRadius or CellSlopes is null')
        return 0
    self.cell_velocs = np.power(n, -1) * np.power(cell_hradius, 2 / 3) * np.sqrt(cell_slopes)
    return self.cell_velocs

'''
计算每个单元格到出口点所使用的时间
---------------
初始化参数
array_velocity:流量速度阵列
array_fdir:方向阵列数据
return:流域范围内每个单元格到出口点的时间多维数组
---------------
'''

def getFlowTime(self, array_velocity=None, array_fdir=None, coordinate_x=None, coordinate_y=None, row=None,
                column=None):
    ...
    array_travel_time = np.zeros((len(array_fdir), len(array_fdir[0])), dtype=np.float64)
    # 创建存储节点链路关系数据，用于存储任意单元到出口点的链路关联信息
    array_node_link = []
    # 出口点对应的栅格行列坐标信息
    outlet_row = -1
    outlet_column = -1

    # 获取2个单元格间的直线距离
    cell_size = self.getCellSize()
    # 获取2个单元格之间的斜线距离
```

输出成果

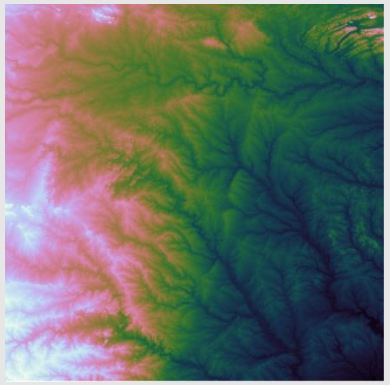


子流域范围

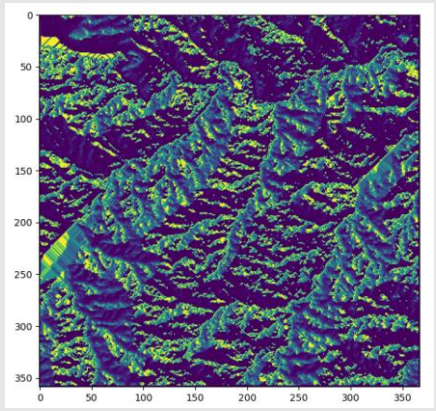


汇流单位线

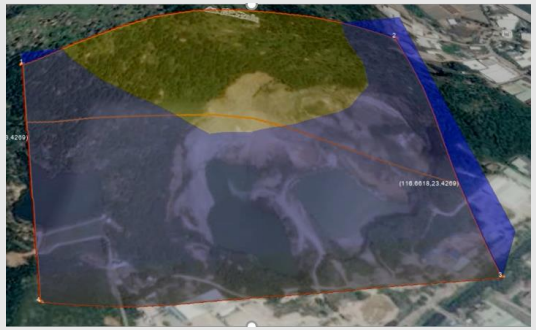
堰塞湖库容曲线提取模块--算法流程



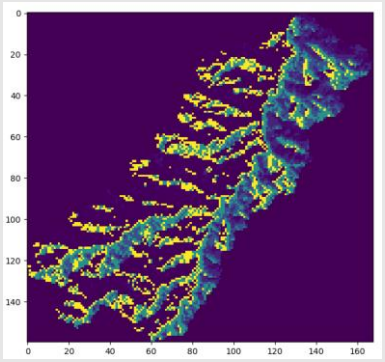
1、数字高程模型生成及获取



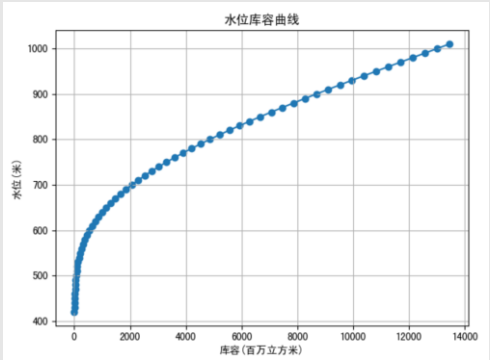
2、选定出水口，提取库区DEM



3、计算不同蓄水位的淹没区域DEM



4、以淹没区域DEM为基础计算蓄水位高程以下库容



5、库容曲线生成

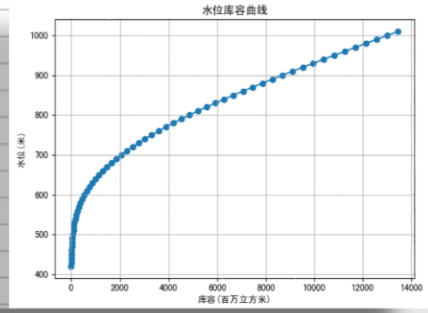
堰塞湖库容曲线提取模块--关键代码及输出

关键代码

```
'''
根据坝体高程对裁剪后的栅格数据进行代数运算，提取高程以下的栅格数据
----------
初始化参数
input_raster: 输入栅格数据
output_raster: 输出栅格数据
----------
'''

def gdalCalc(self, input_raster=None, output_raster=None, z_value=0):
    script = self.calc_script
    cmd = ['python', script, '-A', input_raster, '--outfile=' + output_raster,
           '--calc=A*(A<=' + str(z_value) + ')', '--NoDataValue=0',
           '--overwrite']
    result = subprocess.call(cmd, close_fds=True)
    return result

'''
根据创建的线程池，把需要并行化的函数传递进来
----------
初始化参数
----------
'''

def runCalcInThread(self):
    for i in range(self.thread_count):
        t = threading.Thread(target=self.gdalCalc)
        self.threads.append(t)
        t.start()

'''
输出对应坝体高程点的水位和库容数据
----------
初始化参数
dam_height: 坝体高度
----------
'''

def getCapacityValue(self, dam_height):
    time_stamp = str(int(time.time()))
    if (self.gdalCalc(self.output_raster, self.output_calc_raster + time_stamp + '.tif', dam_height) == 0):
        return self.gdalRead(self.output_calc_raster + time_stamp + '.tif')

'''
输出库容曲线信息
```

输出成果

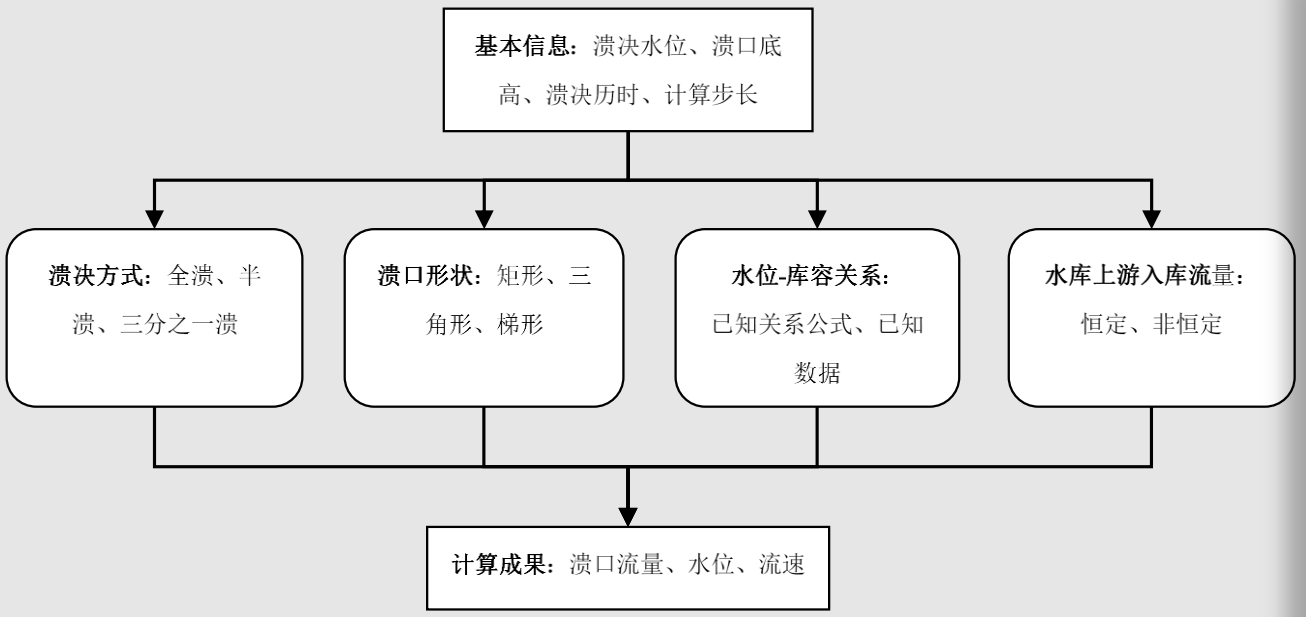
	z	c
2	420	0
3	423	0.013552009
4	426	0.09769623
5	429	0.249065225
6	432	0.466205192
7	435	0.76990834
8	438	1.194625861
9	441	1.77208513
10	444	2.521924866
11	447	3.445015833
12	450	4.527862428
13	453	5.754920836
14	456	7.135675655
15	459	8.698203795
16	462	10.48861825
17	465	12.54177823
18	468	14.87798146
19	471	17.4923359
20	474	20.36788052
21	477	23.48272025
22	480	26.86188254
23	483	30.55497154
24	486	34.58761294
25	489	38.99713806
26	492	43.84390781
27	495	49.13579828
28	498	
29	501	
30	504	
31	507	
32	510	
33	513	
34	516	
35	519	
36	522	
37	525	
38	528	
39	531	



堰塞湖库容曲线

溃坝洪水分析模块--算法流程及实现

计算流程



关键代码

```
'''
根据初始化条件及参数计算溃口的时间流量、时间水位、时间库容曲线过程
'''
def calculation(self):
    retu_t = []
    retu_z = []
    retu_q = []
    retu_cp = []

    this_time = 0
    this_z = self.__z
    this_cp = self.get_relation_cp(self.__z)
    for index in range(self.get_interval_num() + 1):
        this_time += self.__time_interval
        this_deep = this_time / self.__duration * self.__max_deep - (self.__z - this_z)
        this_wide = self.__max_wide
        if self.__shape == self.SHAPE_TRIANGLE:
            this_wide = (this_deep / math.tan(math.radians(self.__dip_angle_left))) + (
                        this_deep / math.tan(math.radians(self.__dip_angle_right)))
        if self.__shape == self.SHAPE_TRAPEZOID:
            this_wide = self.__max_wide - (math.tan(math.radians(self.__dip_angle_left)) + math.tan(
                math.radians(self.__dip_angle_right))) / (math.tan(math.radians(self.__dip_angle_left)) * math.tan(
                math.radians(self.__dip_angle_right))) * (self.__max_deep - this_deep)
        this_q = math.pow(((2 * self.__N + 2) / (2 * self.__N + 3)), (2 * self.__N + 3)) / math.pow((self.__N + 1),
                                                                                                     (
                                                                                                             3 / 2)) * math.pow(
            self.__G, 0.5) * this_wide * math.pow(this_deep, (3 / 2))
        temp_time_interval = self.__duration % self.__time_interval if self.get_interval_num() == index else self.__time_interval
        if temp_time_interval == 0:
            continue
        this_cp = this_cp - this_q * temp_time_interval * 60
        this_z = self.get_relation_z(this_cp)

        retu_t.append(this_time)
        retu_z.append(this_z)
        retu_q.append(this_q)
        retu_cp.append(this_cp)
```

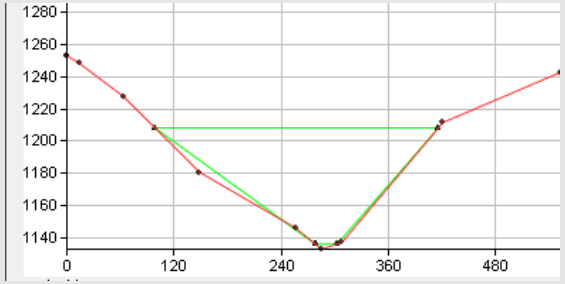
关键算法

$$Q_{\max}=\frac{\left(\frac{2n+2}{2n+3}\right)^{2n+3}}{(n+1)^{\frac{3}{2}}}\sqrt{g}B_{top}H_0^{\frac{3}{2}}$$

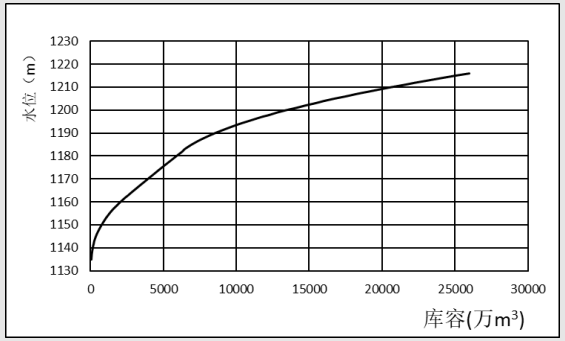
林秉南-龚振赢-王连祥-路吉康公式

溃坝洪水分析模块

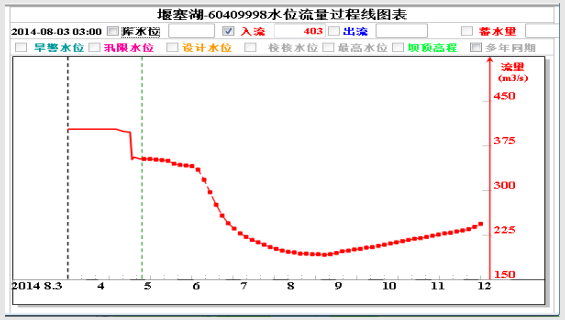
输入数据



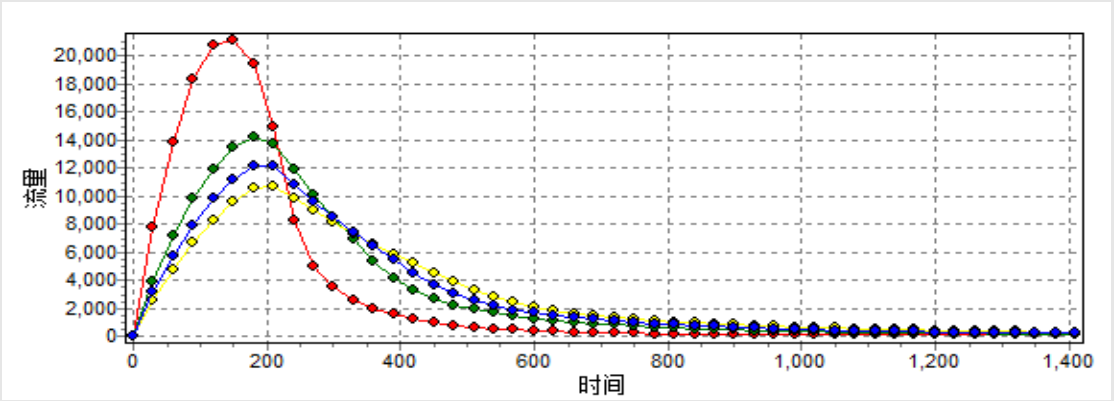
实测断面信息



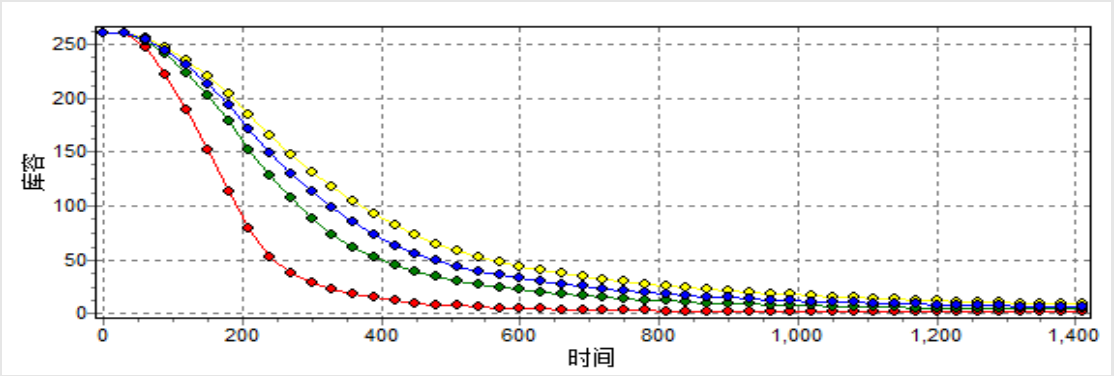
堰塞湖库容曲线



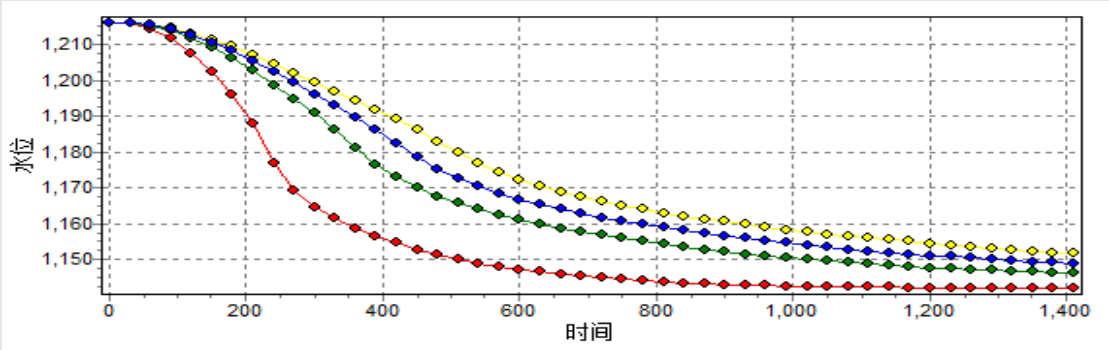
来水预测



溃口流量过程



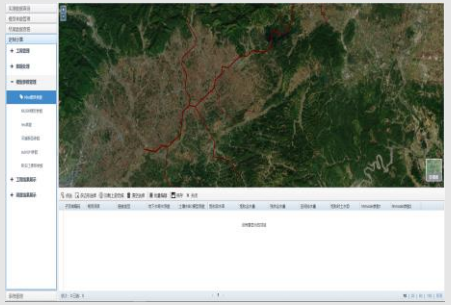
堰塞湖库容变化过程



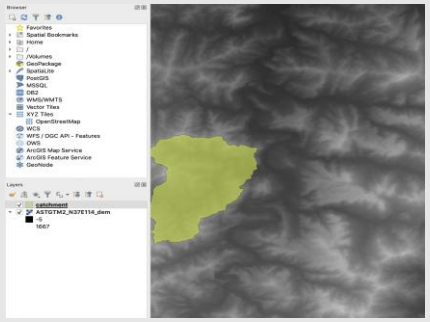
堰塞湖水位变化过程

输出数据

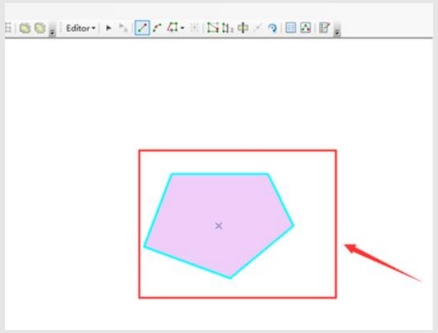
一维河道淹没快速分析模块--算法流程



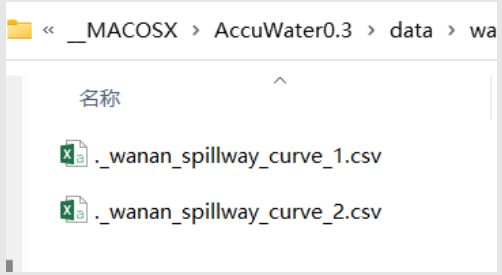
初始化AWCrossSection主类对象及相关属性信息



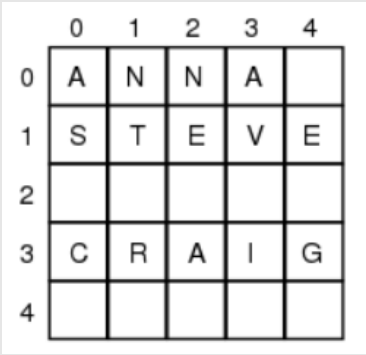
求断面线图层与边界图层的交集结果图层



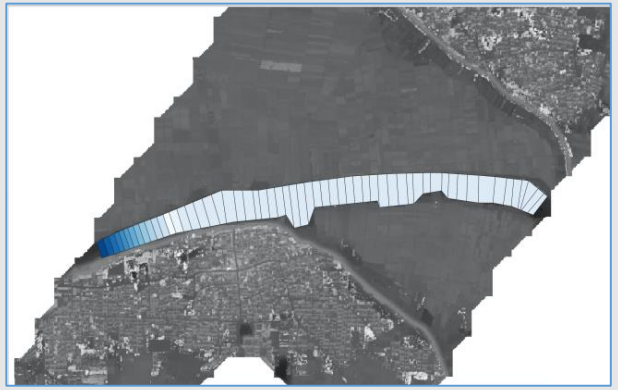
根据裁剪后的断面线图层，将相邻的断面线组合生成闭合的断面polygon实体矢量对象



计算获取各个断面间的间距信息，并写入指定的csv文件中



计算输出不同断面位置的起点距和高程数组



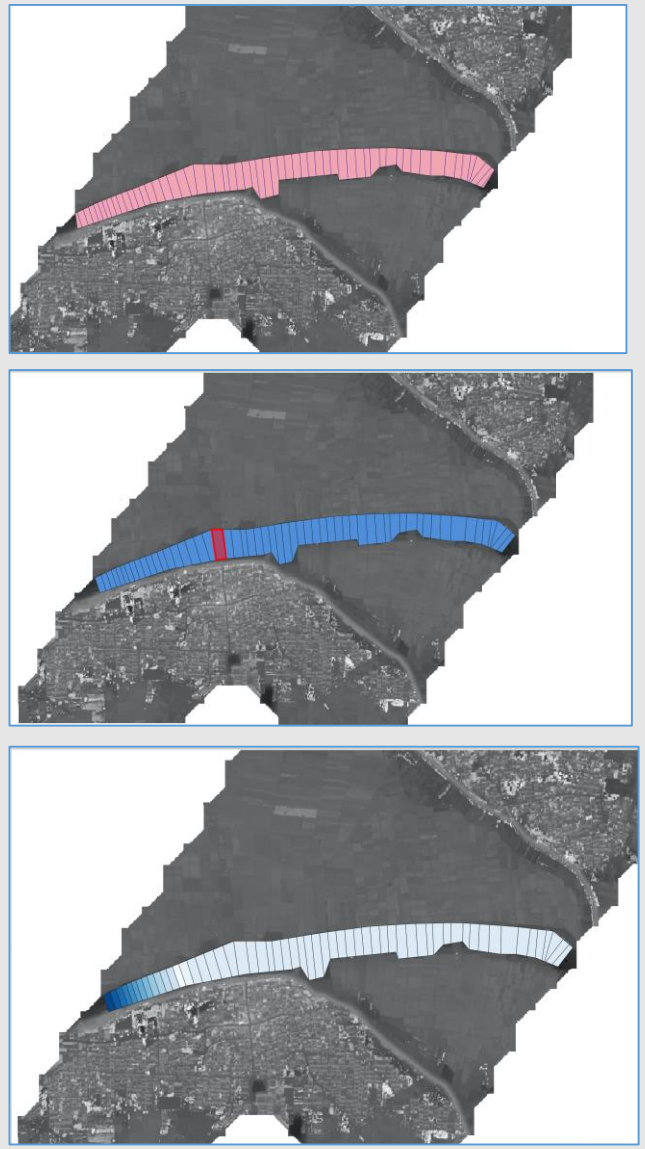
调用不规则断面一维河道水面线计算程序，生成不同断面的水位、流量数据

一维河道淹没快速分析模块--关键代码及输出

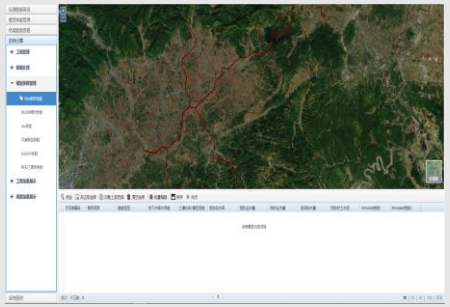
关键代码

```
'''
accu_1D_func
不规则断面一维河道水面线计算程序
'''

def accu_1D_func(self, isCreat=True):
    if isCreat:
        n = self.n1 - 1  # 断面总数
        N = self.n1 * 2
        t = int(self.T / self.dt)  # 时间段数
        x = np.zeros(N)  # x 为Zi，Qi的解向量

        # B: 五对角矩阵方程的常数项，I,J,K,M,O:五对角矩阵A的五列
        B = np.zeros(N)
        I = np.zeros(N)
        J = np.zeros(N - 1)
        K = np.zeros(N - 2)
        M = np.zeros(N - 1)
        O = np.zeros(N - 2)
        # Z，Q分别为模拟时段内的水位和流量过程，X：水位、流量组合矩阵
        Z = np.zeros((t + 1, n + 1))  # 最后整理结果时用
        Q = np.zeros((t + 1, n + 1))  # 最后整理结果时用
        X = np.zeros((N, t + 1))

        # x 解向量初始化
        for i in range(0, N, 2):
            x[i] = self.Z0
            x[i + 1] = self.q0
        # 上下边界水位初始化。第一个从0开始，最后一个是N-1，最后一个水位是N-2.和matlab太不一样了
        x[0] = self.Zu
        x[N - 2] = self.Zd

        # 将x付给X的第一列? matlab程序做了转置，这里貌似不需要
        X[:, 0] = x
        # print(X) # 结果确实是第0列发生了改变，因此无需转置的做法。

        # 初始化B,I,J,M向量
        B[0] = self.Zu
        B[N - 1] = self.Zd
        I[0] = 1
        I[N - 1] = 0
        J[0] = 0
        M[N - 2] = 1

        # 读取断面间距 csv文件
```

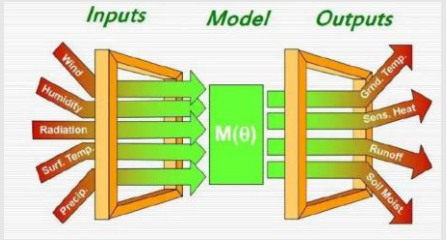
输出成果



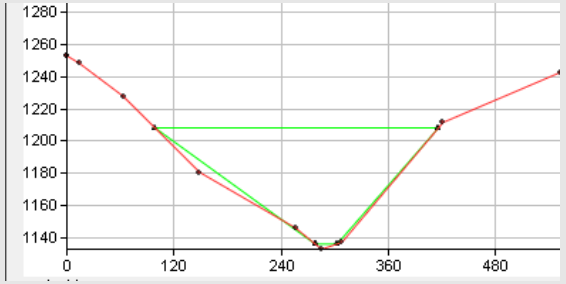
二维区域淹没快速分析模块--算法流程



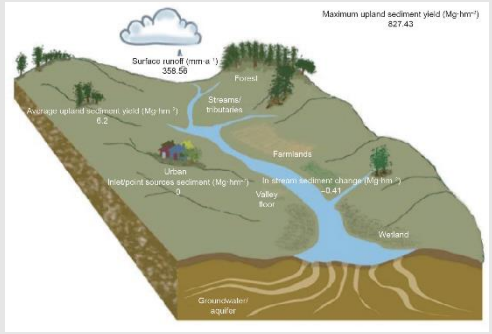
初始化AWFlood主类及默认参数信息



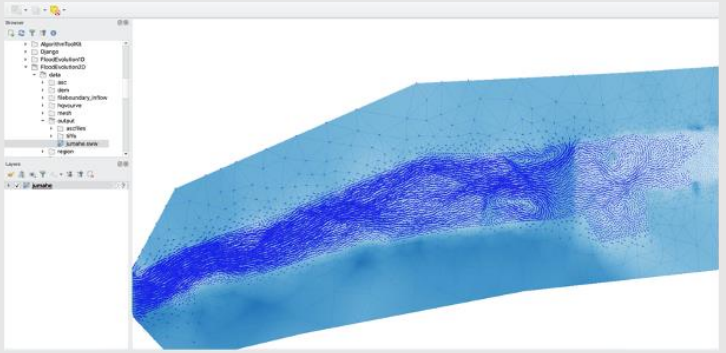
根据初始化参数设置模型运行相关条件



设定边界条件



根据设定好的边界条件，开始模拟计算

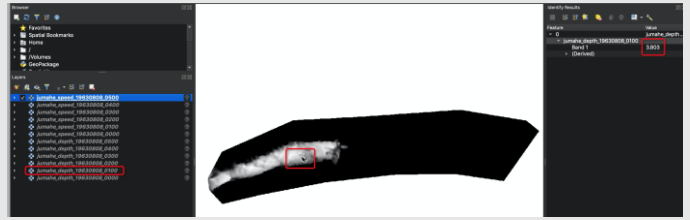


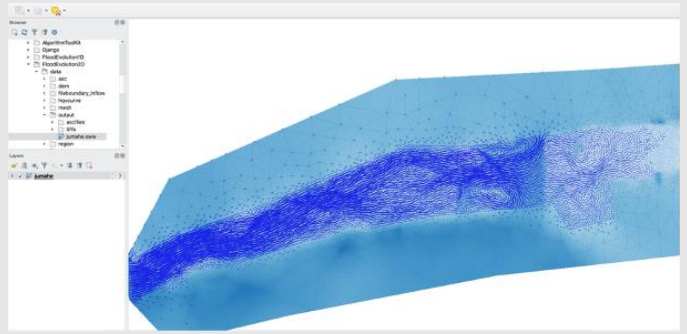
输出不同时刻的淹没范围、淹没深度

二维区域淹没快速分析模块--关键代码及输出

关键代码

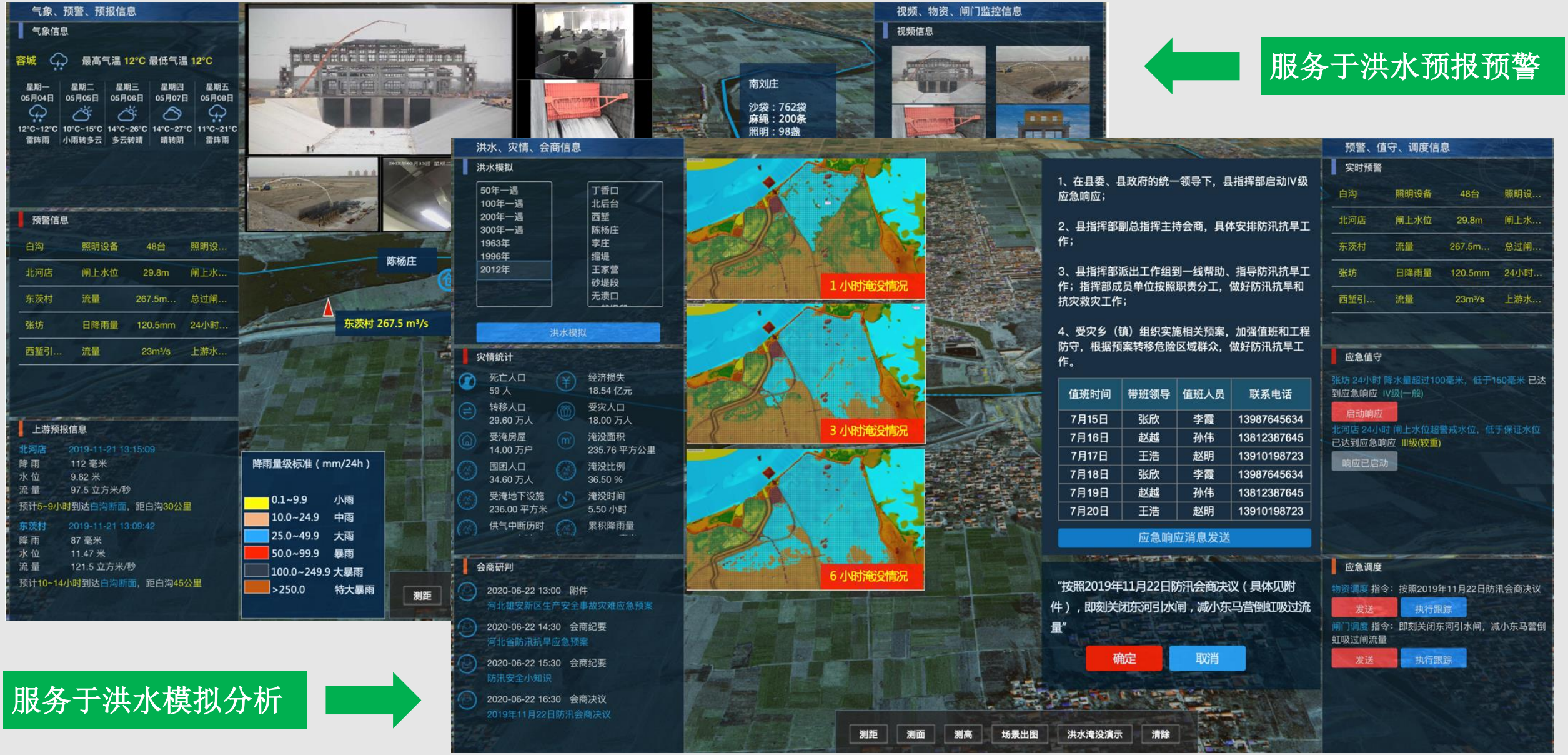
```
'''
基于Anuga的二维区域淹没快速分析模块
---------------
初始化参数
river_friction:糙率
yieldstep:时间步长
finaltime:总时长
inflow_csv:文件型入流边界：时间流量数据
stage:初始化水位
outflow:出流边界水位
re_reasasc:是否重新生成栅格文件
re_create_mes:是否重新生成网格文件
default_res:默认外部区域网格分辨率
inner_res0:默认内部区域网格分辨率
boundary_tags_in0:入流边界标识
boundary_tags_out0:处理边界标识
---------------
'''

class AWFlood:
    def __init__(self, river_friction=0.01, yieldstep=600, finaltime=3600,
                 inflow_csv=None,
                 stage=2700, outflow=2650, re_reasasc=True, re_create_mesh=True,
                 default_res=100, inner_res0=20, boundary_tags_in0=0, boundary_tags_out0=21):...

    '''...'''

    def set_conditions(self):...

    '''
    设定边界条件，优先选择QTime_boundary或者QFile_boundary作为入流断面的边界条件
    定义边界条件，anuga能够定义多种边界条件，根据情况选择"""
    狄利克雷边界（入流），适用于固定入流或者出流
    '''

    def set_boundaries(self):
        # inflow，三个参数分别是stage，x，y方向的动量。动量在这里是速度和水深的乘积。也就是单宽流量，一般取0。
        self.bin_stage = anuga.Dirichlet_boundary([self.stage, 0, 0])

        # 反射墙边界（阻挡）
        # Solid reflective wall，固定反射墙，暂时用于非入流，出流边界
        self.b_reflect = anuga.Reflective_boundary(self.domain)

        # 狄利克雷边界（出流）适用于固定入流或者出流
        # outflow，出流边界，设定水位较低方便出流
```





输出成果

应用案例--雄安新区南拒马河防洪信息管理系统



服务于洪水预报预警

服务于洪水模拟分析

水文风险模拟分析子系统

命令行操作流程

目录结构

脚本文件

参数设置

命令提交

运行过程

结果查看

分系统目录： /data/luoyong/H03

子系统目录：

算法脚本目录：/data/luoyong/H03/algorithm/SWFX

输入数据目录：/data/luoyong/H03/inputdata/SWFX

输出结果目录：/data/luoyong/H03/product/SWFX

模块目录：

算法脚本目录：/data/luoyong/H03/algorithm/SWFX/${模块缩写}

输入数据目录：/data/luoyong/H03/inputdata/SWFX/${模块缩写}

输出结果目录：/data/luoyong/H03/product/SWFX/${模块缩写}

模块	缩写
子流域划分及汇流单位线提取模块	ZLHF
堰塞湖库容曲线提取模块	YSHK
溃坝洪水分析模块	KBHS
一维河道淹没快速分析模块	YWHD
二维区域淹没快速分析模块	EWQY

子系统运行脚本：

提交脚本：run_SWFX.sh

```
#!/usr/bin/bash
source ~/.bashrc
conda activate swfx
# 判断是否传递用户配置参数文件,不存在则终止程序
if [ -z $1 ];then
    echo "未输入参数配置文件userconfig.json"
    exit 1
fi

# 程序记录开始时间
starttime=$(date '+%Y-%m-%d %H:%M:%S')
MODULE="SWFX"
MODULENAME="水文风险分析"

#接收日志文件输出路径
log_file=$(cat $1 | jq .algorithmLogFile -r)
json_file=$(cat $1 | jq .resultJsonFile -r)
flow_file=$(cat $1 | jq .algorithmFlowFile -r)


# 检查日志和json文件是否已经存在，存在则删除
if [ -f $log_file ];then
    rm $log_file
fi
if [ -f $json_file ];then
    rm $json_file
fi
if [ -f $flow_file ];then
```

```
LOG "执行子流域划分主算法 AWSubwater_Unit.py"
cd ${ZLHFPATH}
run_python AWSubwater_Unit $ZLHFPATH_USERCONF 2

LOG "执行库容曲线提取主算法 AWGDAL_Unit.py"
cd ${YSHKPATH}
run_python AWGDAL_Unit $YSHKPATH_USERCONF 3

LOG "执行库容曲线插值主算法 AWInterpolation_Unit.py"
cd ${ATKPATH}
run_python AWInterpolation_Unit $ATKPATH_USERCONF 4

LOG "执行汇流单位线提取主算法 AWIUH_Unit.py"
cd ${ZLHFPATH}
run_python AWIUH_Unit $ZLHFPATH_USERCONF 5

LOG "执行溃坝分析主算法 AWDamBreak_Unit.py"
cd ${KBHSPATH}
run_python AWDamBreak_Unit $KBHSPATH_USERCONF 6

LOG "执行一维河道分析主算法 AWFloodOneD_Unit.py"
cd ${YWHDPATH}
run_python AWFloodOneD_Unit $YWHDPATH_USERCONF 7


LOG "执行二维淹没分析分析主算法 AWFlood2D_Unit.py"
cd ${EWQYPATH}
run_python AWFlood2D_Unit $EWQYPATH_USERCONF 8
```

分析模块运行提交：

bash run_SWFX.sh useconfig.json

分析模块运行：

屏幕实时打印结果

```
(base) [luoyong@server02 SWFX]$ ./run_SWFX.sh userconfig.json
2022-05-27 10:48:04 : 主程序初始化
2022-05-27 10:48:04 : 完成参数解析
2022-05-27 10:48:05 : 执行子流域划分主算法 AWSubwater_Unit.py
2022-05-27 10:49:03 : 主程序算法 AWSubwater_Unit.py 执行完毕,耗时0m57.915s
2022-05-27 10:49:03 : 执行库容曲线提取主算法 AWGDAL_Unit.py
2022-05-27 10:49:46 : 主程序算法 AWGDAL_Unit.py 执行完毕,耗时0m43.160s
2022-05-27 10:49:46 : 执行库容曲线插值主算法 AWInterpolation_Unit.py
2022-05-27 10:49:51 : 主程序算法 AWInterpolation_Unit.py 执行完毕,耗时0m4.852s
2022-05-27 10:49:51 : 执行汇流单位线提取主算法 AWIUH_Unit.py
2022-05-27 10:50:00 : 主程序算法 AWIUH_Unit.py 执行完毕,耗时0m8.219s
2022-05-27 10:50:00 : 执行溃坝分析主算法 AWDamBreak_Unit.py
2022-05-27 10:50:03 : 主程序算法 AWDamBreak_Unit.py 执行完毕,耗时0m3.228s
2022-05-27 10:50:03 : 执行一维河道分析主算法 AWFloodOneD_Unit.py
2022-05-27 10:50:13 : 主程序算法 AWFloodOneD_Unit.py 执行完毕,耗时0m9.729s
2022-05-27 10:50:13 : 执行二维淹没分析分析主算法 AWFlood2D_Unit.py
2022-05-27 10:51:18 : 主程序算法 AWFlood2D_Unit.py 执行完毕,耗时1m4.565s
2022-05-27 10:51:18 : 输出产品路径 /data/luoyong/H03/product/SWFX/
2022-05-27 10:51:18 : 输出json路径 /data/luoyong/H03/product/SWFX/output.json
2022-05-27 10:51:18 : 输出日志路径 /data/luoyong/H03/product/SWFX/output.log
2022-05-27 10:51:18 : 输出算法执行流程路径 /data/luoyong/H03/product/SWFX/outputFlow.json
```

汇报完毕，谢谢