

# 土木研究所資料

## 降雨流出解析と連動した 土石流の流出・氾濫解析法

令和4年3月

国立研究開発法人土木研究所  
土砂管理研究グループ火山・土石流チーム

Copyright © (2022) by P.W.R.I.

All rights reserved. No part of this book may be reproduced by any means, nor transmitted, nor translated into a machine language without the written permission of the Chief Executive of P.W.R.I. The program source code is licensed under CC BY-SA (Attribution-ShareAlike) 4.0.

この報告書は、国立研究開発法人土木研究所理事長の承認を得て刊行したものである。したがって、本報告書の全部又は一部の転載、複製は、国立研究開発法人土木研究所理事長の文書による承認を得ずしてこれを行ってはならない。ただし、プログラムソースコードのライセンスは CC BY-SA (表示-継承) 4.0 である。

# 降雨流出解析と連動した 土石流の流出・氾濫解析法

火山・土石流チーム	元研究員	山崎祐介
火山・土石流チーム	研究員	清水武志
火山・土石流チーム	前上席研究員	石井靖雄
火山・土石流チーム	上席研究員	石田孝司

## 要 旨：

土石流による被害の防止・軽減のためには、土石流の発生時期や氾濫範囲を推定する必要がある。近年、降雨や空中写真、標高データなどの時間・空間分解能が高く、かつ、リアルタイム性の高いデータを得られる環境が整いつつあることから、火山・土石流チームでは、土石流の氾濫危険度評価のさらなる精度向上を図るための土石流流出・氾濫モデルの開発に取り組んでいる。本資料では、開発したモデルの説明と実際の土石流への適用事例を紹介する。また、解析プログラムソースコードと計算に用いたデータを添付する。

キーワード：降雨流出、土石流氾濫、数値解析

## 目次

1. はじめに.....	1
1.1. 背景と目的 .....	1
1.2. 本資料の構成.....	2
1.3. 本解析モデルの特徴.....	3
1.3.1. 土石流の流量の形成過程.....	3
1.3.2. 氾濫条件の設定 .....	4
1.4. 本解析モデルの課題.....	5
2. 解析モデル.....	6
2.1. 解析モデルの概要 .....	6
2.2. モデル分割の目的 .....	9
2.3. 降雨流出モデル.....	10
2.3.1. 斜面における浸透流・洪水流の支配方程式.....	10
2.3.2. 河道への流出量の算出方法 .....	12
2.3.3. 河道における洪水流の支配方程式 .....	12
2.3.4. 離散化 .....	13
2.4. 土石流流出モデル .....	16
2.4.1. 河床堆積物の安定条件 .....	17
2.4.2. 土石流の支配方程式.....	17
2.4.3. 土石流の溢水量の算出方法 .....	19
2.4.4. 離散化 .....	20
2.5. 土石流氾濫モデル .....	24
2.5.1. 土石流の支配方程式.....	24
2.5.2. 離散化 .....	25
3. モデルの適用事例.....	30
3.1. 降雨流出.....	31
3.1.1. 解析モデル.....	31
3.1.2. 計算に用いたデータと条件 .....	31
3.1.3. 計算結果 .....	31
3.2. 土石流流出.....	34
3.2.1. 桜島有村川における土石流観測.....	34
3.2.2. 解析モデル.....	39
3.2.3. 計算に用いたデータと条件 .....	39
3.2.4. 計算結果 .....	40
3.3. 土石流氾濫 .....	42

3.3.1. 阿蘇古恵川の土石流災害の概要.....	42
3.3.2. 解析モデル.....	42
3.3.3. 計算に用いたデータと条件.....	44
3.3.4. 計算結果.....	46
4. おわりに.....	49
A. 付録.....	A-1
A.1. 寺内ダムの計算セット「Terauchi_Dam」.....	A-2
A.1.1 「01_SET_BASIN」フォルダ.....	A-2
A.1.2 「02_MK_TOPO」フォルダ.....	A-3
A.1.3 「03_MK_RAIN」フォルダ.....	A-4
A.1.4 「04_EXE_SIMU」フォルダ.....	A-5
A.1.5 「05_MK_FIG」フォルダ.....	A-6
A.2. 桜島有村川の計算セット「Arimura_Riv」.....	A-7
A.2.1 「01_SET_BASIN」フォルダ.....	A-7
A.2.2 「02_MK_TOPO」フォルダ.....	A-8
A.2.3 「03_MK_RAIN」フォルダ.....	A-9
A.2.4 「04_EXE_SIMU」フォルダ.....	A-10
A.2.5 「05_MK_FIG」フォルダ.....	A-10
A.3. 阿蘇古恵川の計算セット「Furuegawa_Riv」.....	A-11
A.3.1 「01_SET_BASIN」フォルダ.....	A-11
A.3.2 「02_MK_TOPO」フォルダ.....	A-12
A.3.3 「03_MK_RAIN」フォルダ.....	A-14
A.3.4 「04_EXE_SIMU」フォルダ.....	A-15
A.3.5 「05_MK_FIG」フォルダ.....	A-17
B. 付録	
プログラムソースコード	

## 1. はじめに

### 1.1. 背景と目的

土石流による被害の防止・軽減のためには、土石流の発生時期や氾濫範囲の推定が重要である。天然ダムの形成や火山の噴火に伴う降灰が確認された場合には、改正土砂災害防止法により、国土交通省が土石流の被害が想定される区域を示す必要がある<sup>1)</sup>。そのため、数値シミュレーション手法に基づく QUAD モデル (Quick Analysis System for Debris Flow) が開発された。しかし、このモデルは火山噴火直後に迅速に解析結果を得ることに主眼を置いているため、ある特定の状況のシミュレーションに機能が限定される<sup>2)</sup>。一方、無償で利用できる汎用的な土石流の数値シミュレーションソフトウェアも公開されており、例えば、高橋らの研究成果<sup>3) 4) 5)</sup>に基づいた Kanako モデル<sup>6) 7) 8) 9)</sup> (Kanako 1D および Kanako 2D を示す) や江頭らの研究成果<sup>10),11)</sup>に基づいた iRIC のソルバーである Morpho2DH モデル<sup>12)</sup><sup>13)</sup> が知られている。

近年では、雨量や空中写真、標高などの時間・空間分解能が高く、かつ、リアルタイム性の高いデータを得られる環境が整いつつある<sup>14)</sup> ことから、火山・土石流チームでは、土石流氾濫危険度のさらなる評価精度の向上を図るための土石流流出・氾濫モデルの開発に取り組んでいる。本資料では、開発したモデル (以降、本解析モデルという) の説明と実際の土石流への適用事例を紹介するとともに、計算に用いたデータおよびプログラムソースコードを添付する。なお、本稿では、水および土砂の混合物、例えば土石流、泥石流、土砂流や、流砂をとまなう洪水流などを総称して土石流という。

解析プログラム、入出力データ処理プログラム、および可視化プログラム群は「Debris Flow Simulator for Sabo (DFSS)」と称する。「DFSS」の著者は土木研究所である。ライセンスは、「政府標準利用規約 (第 2.0 版)」を参考に、クリエイティブ・コモンズ表示-継承 4.0 国際パブリック・ライセンス (以下 CC BY-SA 4.0) とする。したがって、利用者は、ソースコードの利用にあたり、次の 2 点を承諾し、遵守しなければならない。

1.著作権表示

2.継承 (2 次著作物を作った場合、CC BY-SA 4.0 のライセンスを付与)

また、利用者はいかなる利用に対しても以下の点を承諾したものとする。

1.利用者は商用利用や改変も含めて自由に利用できること

2.利用者の利用に対して、著作者および関係者は一切その責任を負わないこと

3.著作者および関係者は一切のサポートを行わないこと

## 1.2. 本資料の構成

本資料の構成は、次に示すとおりとする。

1章では、本解析モデルを開発した背景と目的、本解析モデルの特徴、本解析モデルの今後の課題をまとめた。本解析モデルの特徴として、既往の土石流のモデルにおける、土石流の流量の形成過程の扱いと氾濫条件の設定方法の考え方を比較しながら、本解析モデルが採用した方法の特徴を概説する。

2章では、本解析モデルを構成している、降雨流出モデル、土石流流出モデル、および土石流氾濫モデルのそれぞれについての基礎式およびその離散化について、詳細を説明している。

3章では、実際に発生した降雨流出、土石流流出、および土石流氾濫について、本解析モデルを適用し、その結果を示している。

4章では、3章の結果から、本解析モデルの適用性を評価している。

5章では、3章で行った計算のフォルダ構成、解析プログラム名、データ処理プログラム名、入力ファイル名、および出力ファイル名を示している。

6章では、プログラム一式のソースコードを示している。

### 1.3. 本解析モデルの特徴

土石流の氾濫範囲は、氾濫地形および土石流の氾濫条件、つまり、氾濫発生位置と、その地点における土石流の流量によって概ね決定される。これらを精度良く推定するためには、土石流の発生・流下過程を解析することが重要である。本解析モデルの特徴は、土石流の発生・非発生については、降雨条件が重要な役割を果たしていることを考慮して、降雨流出過程、土石流の発生・流下過程、および土石流の氾濫過程を扱うところにあり、そのため、降雨流出モデル、土石流流出モデル、土石流氾濫モデルから構成されている。土石流の発生から流下過程における流量の形成過程と氾濫条件の設定については、モデルによって考え方が異なることもある。よって、ここでは既往のモデル（QUAD、Kanakano、Morpho2DH）と本解析モデルにおける違いを示す。

#### 1.3.1. 土石流の流量の形成過程

土石流の発生から流下過程における流量の形成過程は、崩壊により生産された崩土の移動過程における流動化、崩土により形成された河床堆積物（河道閉塞）あるいは元から河床に存在していた堆積物が洪水流の侵食によって流れに取り込まれて流動化するものであると考えられているものの、これらは観測に基づいて示されているわけではない。よって、これらの過程によって形成される土石流の流量の扱いについては、前述のモデルによって、次のような違いがみられる。

QUAD では、天然ダムが形成された場合は、湛水状態を初期条件とした越流侵食過程に土石流の支配方程式<sup>15)</sup>を適用して、河床堆積物が洪水流の侵食によって流れに取り込まれて流動するものと考えて、天然ダム天端位置における土石流の流量を定めており、火山の噴火に伴う降灰の場合は、降灰が地表面の浸透能へ及ぼす影響を考慮した降雨流出解析を適用し、得られた流量とその地点の勾配に見合う土砂量を加算して、谷出口における土石流の流量を定めている。Kanakano では、崩壊土量と崩壊継続時間から流量波形を作成し、境界条件として与えている<sup>16)</sup>。Morpho2DH では、崩壊によって生産された流動土砂を初期条件として土石流の支配方程式を適用して定めている。

本解析モデルでは、流域を対象として降雨流出解析<sup>17)</sup>を適用して斜面から河道への降雨流出を算出し、水供給された河床堆積物に無限長斜面の安定解析を適用し<sup>17)</sup>、不安定と判定された部分が流動化するものとして扱い、土石流の支配方程式<sup>10),11),18),19)</sup>を適用して土石流の流量を定めている。ここには、土石流の形成と規模の決定機構に重要な役割を果たす微細砂の流体層への遷移現象<sup>20)</sup>も考慮している。

火山噴火に伴う降灰が生じた直後では、降灰による地表面被覆が浸透能を減少させ、表面流が発生しやすくなり、土石流が発生しやすくなるといわれている。一方、その合理的なモデルは示されていないため、本解析モデルでは、浸透と降灰厚を関係づけてよいかは不明であるが便宜上、降灰後の各位置の浸透能は降灰厚を変数とする 1 次関数に従って減少するモデルを仮定して定めている。



### 1.3.2. 氾濫条件の設定

QUAD の土石流の氾濫計算では、谷出口を氾濫開始地点として設定し、そこに土石流の流量波形を境界条件として与える方法が用いられている。したがって、谷出口周辺の土石流の流動の計算結果は、境界条件の影響を受ける。Kanako (Kanako 2D) では、1次元領域 (Kanako 1D) を2次元領域まで延伸し、重なっている部分において土石流の侵食・堆積を伴う流動の相互作用を評価している。Morpho2DH では、計算領域の全体を2次元平面で計算しているため、氾濫開始点の設定についての問題は生じない。

本解析モデルでは、降雨流出モデルと土石流流下モデルは同じ空間に設定され、土石流氾濫モデルは、別の空間に設定される。これらの空間の重なっている部分において、土石流流下モデルで計算された土石流の表面標高が地盤標高を上回った地点では、土石流の水深に越流公式を適用して土石流の溢流量を算出し、これを土石流氾濫モデルに境界条件として与えている。本解析モデルでは、氾濫発生地点は、土石流が河道を流下していく過程で土石流の表面標高と流路側岸の地盤標高によって定まるため、条件を満たす氾濫発生地点が計算過程で定められていく。一方、各氾濫発生地点で計算される溢流量は、境界条件として氾濫モデルへ与えられるため、土石流流下モデルと土石流氾濫モデル間の相互作用は考慮していない。

#### 1.4. 本解析モデルの課題

本解析モデルにおいては、土石流の氾濫範囲推定に重要な過程を既往研究に基づき、できる限りモデル化するように努めたが、次の点を考慮しておらず課題として挙げられる。

- ・ 山腹崩壊により生産される土砂の流出（崩土の流出）
- ・ 河床堆積物における微細砂含有率の時空間変化の評価
- ・ 溢流時の土石流の流れの鉛直構造
- ・ 土石流対策施設の効果評価

崩土の流出において、当該豪雨により発生した崩壊による生産土砂（崩土）は、地域によっては微細砂を多く含有していることが考えられ、微細砂の供給源として重要である。崩壊の発生については、本稿で用いている降雨流出解析と無限長斜面の安定解析を組み合わせた方法の他、過去の崩壊データと地形データなどの統計解析に基づいた方法<sup>21)</sup>などを用いることができる。崩壊によって生産された崩土の流出については、崩壊発生から土石流が形成される過程を理論的に定める方法、崩土の侵食・堆積をとまなう流出過程に質点系方程式を適用する方法<sup>23)</sup>、および崩壊土量と崩壊継続時間から流量波形を作成し、土石流の境界条件として設定する方法<sup>16)</sup>がある。

河床堆積物は、当該豪雨以前に形成されたものと当該豪雨によって発生した崩土の流出過程において形成されるものから構成される。当該豪雨以前に形成された堆積物は、流水作用により微細砂が流亡していることが考えられる。一方、当該豪雨で形成された堆積物には、微細砂が多く含有していることも考えられる。土石流の侵食により、流れに取り込まれた微細砂は、流体相に遷移し、間隙流体の質量密度を増加させるとともに固相の濃度を減少させるため、土石流の流動性に及ぼす影響が大きい<sup>20)</sup>。よって、河床堆積物における微細砂の時空間変化を評価することが重要である。これには、移動床水理学でよく用いられている、多層モデル<sup>24)</sup>を適用することが有効であると考えられる。

谷出口などの勾配が小さい地点においては、固相を形成する粗粒砂が流れの表面まで分布しない場合がある。このような場合、溢流する大部分は、水と微細砂から形成される流体であると考えられる。粗粒砂の分布範囲（粒子流動層厚）の評価は、土石流の構成則から導くことができる<sup>10),11)</sup>。粗粒砂の質量保存則は、このことを考慮して変更する必要がある。

砂防堰堤などの土石流対策施設の効果評価において、水深平均の支配方程式を適用する場合、砂防堰堤直上流地点の流れには適用できない場合が考えられる。このような場合には、便宜的に、計算点の空間配置を変更<sup>7)</sup>したり、堰堤を越流する際の境界条件を設定<sup>7)</sup>したりする方法を取り入れることが考えられる。

## 2. 解析モデル

### 2.1. 解析モデルの概要

本解析モデルは、降雨により流域斜面に飽和側方浸透流（以降浸透流という）・表面流が発生し、これらが河道に流出し、河床堆積物の不安定化により土石流が発生・流下し、土石流の氾濫が発生する現象を表現したものである。本解析モデルは、地形モデル、降雨流出モデル、土石流流出モデル、土石流氾濫モデルの各サブモデルから構成されている。これらの各サブモデルには、既往の研究成果を活用している。

図 1 に地形モデルを示す。地形モデルは流域地形モデル(□と-)と氾濫原地形モデル(□)から構成されている。流域地形モデルは、斜面系(□)および河道系(-)によって構成され、それぞれ降雨流出モデルおよび土石流流出モデルの計算領域である。斜面系は直交座標上に等分割されたグリッドセルで表現され、河道系は隣接する周囲 8 方向のグリッドセルの中心を結ぶ線分で表現されている。河道の上流端は、ある閾値以上の集水面積を持つグリッドセルであり、最急勾配方向の下流側のグリッドセルに接続していき、下流端に至る。河道の幅・深さは、グリッドセルのサイズから独立して設定することが可能である。氾濫原地形モデルは、等分割されたグリッドセルで表現されている。降雨流出モデルおよび土石流氾濫モデルは平面 2 次元座標を用いて表現され、土石流流出モデルは、1 次元座標を用いて表現されている。

降雨流出モデルは、平面 2 次元で表現された斜面とその上部に設定された表土層において、表土層の湿潤・乾燥、浸透流および表面流の発生・流下過程を計算<sup>17)</sup>する。浸透流および表面流には、それぞれダルシー則およびマニング則が適用されている。河道が設定されているグリッドセルにおいては、グリッドセルの水深に越流公式<sup>25)</sup>を適用して河道への流出量を計算し、これを土石流流出モデルに境界条件として与えている。

土石流流出モデルは、1 次元で表現された河道において、斜面からの浸透流および表面流を境界条件とし、土石流の発生および侵食・堆積をとまなう流下過程を計算する。土石流の発生は、河床堆積物に無限長斜面の安定解析<sup>17)</sup>を適用し、斜面からの流入量により不安定となった場合に、その地点の堆積物の全層が流動化するものとする。流動化した堆積物には、土石流の支配方程式<sup>10),11),18),19)</sup>を適用している。土石流の形成と規模の決定機構に重要な役割を果たす、微細砂の流体相への遷移現象<sup>20),16)</sup>を考慮するため、質量保存則は、固相として流れを支配する粗粒砂と、水とともに流体相として挙動する微細砂に分けて記述されている。土石流の流下過程において、土石流の表面標高が地盤標高を上回った地点では、土石流の水深に越流公式<sup>25)</sup>を適用して土石流の溢流量を算出し、これを土石流氾濫モデルに境界条件として与えている。本解析モデルでは、氾濫発生地点は、土石流が河道を流下していく過程で、土石流の表面標高と流路側岸の地盤標高によって定まる。

土石流氾濫モデルは、河道から溢流した土石流の氾濫過程を計算する。土石流の支配方程式は、平面 2 次元座標における表現であることを除き、土石流流出モデルの支配方程式と同

様である。なお、計算領域は、任意の形状の設定が可能である。

図 2 にサブモデル間の接続関係と入出力項目・設定項目を示す。入力データは、可能蒸発散量（省略可）、降雨データおよび地形データであり、設定項目は、土層・堆積層の水利・水文条件、粒径である。赤い矢印がサブモデル間の接続を示し、降雨流出モデルと土石流流出モデル間では、降雨流出モデルから河道へ浸透流と表面流による流出量が渡され、土石流流出モデルと土石流氾濫モデル間では、土石流流出モデルから氾濫原へ溢流量が渡される。

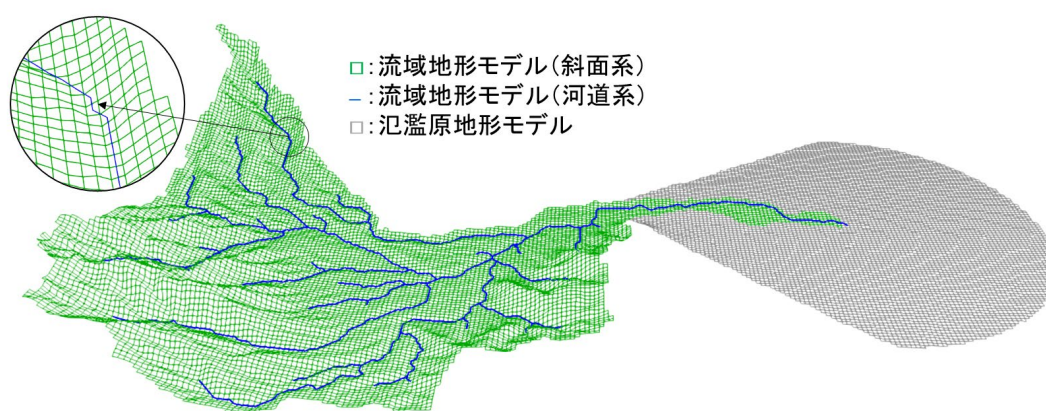


図 1 地形モデルの模式図

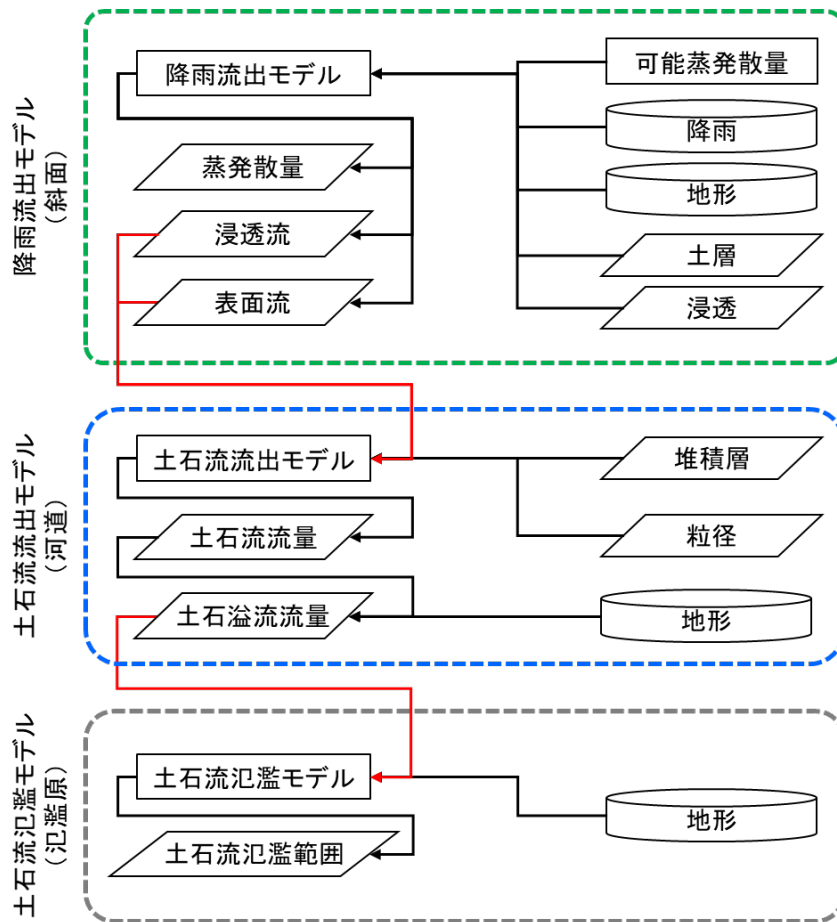


図 2 サブモデル間の接続関係と入出力項目・設定項目

## 2.2. モデル分割の目的

本解析モデルをサブモデルに分割しているのは、計算処理の効率化のためである。降雨流出、土石流流出、および土石流氾濫の計算処理における特徴は以下のとおりである。

- ・ 降雨流出：計算領域が広く、現象の継続時間が長い
- ・ 土石流流出：対象現象の時間変化が大きい
- ・ 土石流氾濫：計算領域の空間解像度が高い

これらを共通のモデルで表現して計算する場合、広い領域を高い空間解像度で表現し、時間刻みを小さくして長い継続時間にわたって計算することになるため、計算資源の消費が大きくなる。本解析モデルは、試行錯誤の計算などを行った上で即時的に結果の概要を得ることについても目的としているため、以上のような計算処理の効率化を図っている。

### 2.3. 降雨流出モデル

降雨流出モデルは、平面 2 次元で表現された斜面とその上部に設定された表土層において、表土層の湿潤・乾燥、浸透流および表面流の発生・流下過程を計算する。図 3 に斜面における表土層、およびそこに形成される浸透流および表面流の水面の模式図を示す。ここに、 $z_s$ および $z$ は表土層の下端および表層の標高、 $D$ は表土層の厚さ、 $\theta$ は最急勾配、 $f_{s1}$ 、 $f_{s2}$ はそれぞれ表層、表層から下層への浸透能、 $p_w$ は表土層の平均水分含有率、 $h_s$ は浸透流水深、 $h$ は表面流水深である。このような斜面に降雨 $r$ が与えられると、雨水の鉛直移動によって表土層が湿潤して $p_w$ が増加し、上限に達すると浸透流が形成され、さらに浸透流水深 $h_s$ が表土層厚 $D$ に達すると、表面流が形成される。降雨が停止すると、蒸発散により浸透流が減少し、表土層が乾燥して $p_w$ が減少する。噴火後の降灰が表土層の透水性に及ぼす影響は、 $f_{s1}$ に反映される。土壌の透水性は、孔隙径分布、孔隙の屈曲度、孔隙同士の連結といった孔隙構造に依存する<sup>26)</sup>が、本研究では簡潔に、 $f_{s1}$ は降灰厚による経験的な関数によって表現されるものとしている。これらの過程について、次のようにモデル化する。

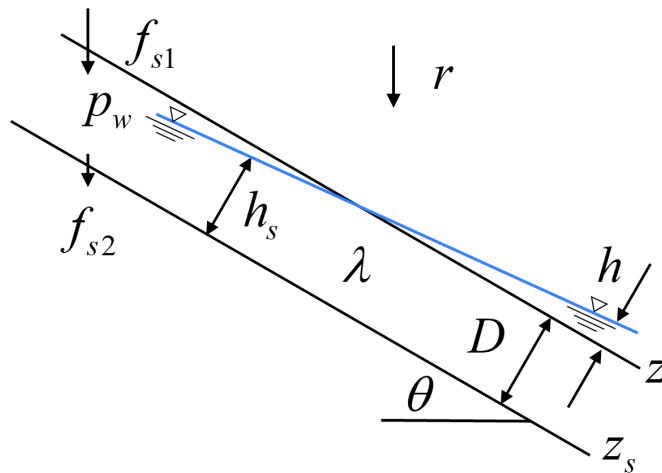


図 3 斜面と水面の模式図<sup>17)</sup>

#### 2.3.1. 斜面における浸透流・洪水流の支配方程式

表土層における水分含有率、浸透流、表面流の質量保存則は、次式で表される<sup>17)</sup>。

$$\frac{\partial(p_w D)}{\partial t} = r - e \quad (1)$$

ここに、 $p_w$ は表土層における水分含有率、 $e$ は蒸発散強度である。 $p_w$ の取り得る最大値を $p_{wc}$ とすれば、同式における $p_w$ の取り得る値は $0 \leq p_w \leq p_{wc}$ であり、 $p_w = p_{wc}$ のとき飽和帯が形成されて浸透流が発生する。降雨が停止すると、蒸発散 $e$ によって乾燥過程に移行する。

浸透流および表面流については、従来の研究においてよく用いられている水深平均の 2 次元支配方程式を採用し、浸透流および表面流にそれぞれダルシー則およびマニング則を適用している。

浸透流の質量保存則は、次のように表される<sup>17)</sup>。

$$\frac{\partial h_s}{\partial t} + \frac{1}{\lambda - p_w} \left( \frac{\partial q_{sx}}{\partial x} + \frac{\partial q_{sy}}{\partial y} \right) = \frac{\cos \theta}{\lambda - p_w} (f_{s1} - f_{s2} - e) \quad (2)$$

ここに、 $t$ は時間、 $x, y$ は直交直線座標、 $x - y$ 面は斜面に沿う平面、 $\lambda$ は表土層の空隙率、 $q_{sx}$ 、 $q_{sy}$ はそれぞれ浸透流フラックス $q_s$ の $x$ 成分および $y$ 成分、 $q_s$ は最急勾配方向の浸透流フラックスである。また、降雨 $r$ が表層の浸透能 $f_{s1}$ を下回る $r < f_{s1}$ のとき、表層の浸透強度は降雨量に等しいから $f_{s1} = r$ とおく。火山地域においては、噴火にともなう降灰により浸透能が小さくなるといわれている。これを表現する関数は不明であるが、便宜的に $f_{s1} = f_{s0}(1 - D_a/D_{a0})$ で表現する。ここに、 $f_{s0}$ は降灰前の浸透能、 $D_a$ は降灰厚、 $D_{a0}$ は浸透能がゼロとなる降灰厚である。

浸透流の運動量保存則は次式で表される<sup>17)</sup>。

$$q_{sx} = -\frac{\partial H}{\partial x} I^{-1} q_s \quad (3)$$

$$q_{sy} = -\frac{\partial H}{\partial y} I^{-1} q_s \quad (4)$$

$$q_s = khI \quad (5)$$

$$I = \sqrt{\left( \frac{\partial H}{\partial x} \right)^2 + \left( \frac{\partial H}{\partial y} \right)^2} \quad (6)$$

ここに、 $H$ は浸透流あるいは表面流の水位で $H = z_s + h_s \cos \theta$ あるいは $H = z + h \cos \theta$ 、 $I$ は水面の最急勾配、 $k$ は飽和透水係数である。

表面流の質量保存則は、次式で表される<sup>17)</sup>。

$$\frac{\partial h}{\partial t} + \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} = (r - f_{s2} - e) \cos \theta - \frac{2q_{sr}L}{\Delta x \Delta y} \quad (7)$$

ここに、 $q_x$ 、 $q_y$ はそれぞれ表面流フラックスの $x$ 成分および $y$ 成分、 $q$ は表面流フラックス、 $q_{sr}$ は斜面から河道への単位幅流量で、負の場合は河道から斜面への流量となる（後述）、 $L$ は単位河道の流下方向の長さであり、流下方向が $x$ 方向のとき $L = \Delta x$ 、 $y$ 方向のとき $L = \Delta y$ である。

表面流の運動量保存則は、次式で表される<sup>17)</sup>。

$$q_x = -\frac{\partial H}{\partial x} I^{-1} q \quad (8)$$

$$q_y = -\frac{\partial H}{\partial y} I^{-1} q \quad (9)$$

$$q = \frac{1}{N} \sqrt{I} h^{5/3} \quad (10)$$

ここに、 $N$ は等価粗度係数<sup>25)</sup>である。



### 2.3.2. 河道への流出量の算出方法

図4に河道が設定されたグリッドセルの模式図を示す。河道の水位がグリッドセルの水位よりも低い場合は、グリッドセルから河道へ流出する。河道の水位がグリッドセルの水位よりも高い場合は、河道からグリッドセルへ流出する。 $q_{sr}$ は河道への単位幅流量であり、完全越流の式<sup>25)</sup>で表すものとする。

$$q_{sr} = \mu h \sqrt{2gh} \quad (11)$$

ここに、 $\mu$ は定数で $\mu = 0.35^{25)}$ 、 $g$ は重力加速度である。

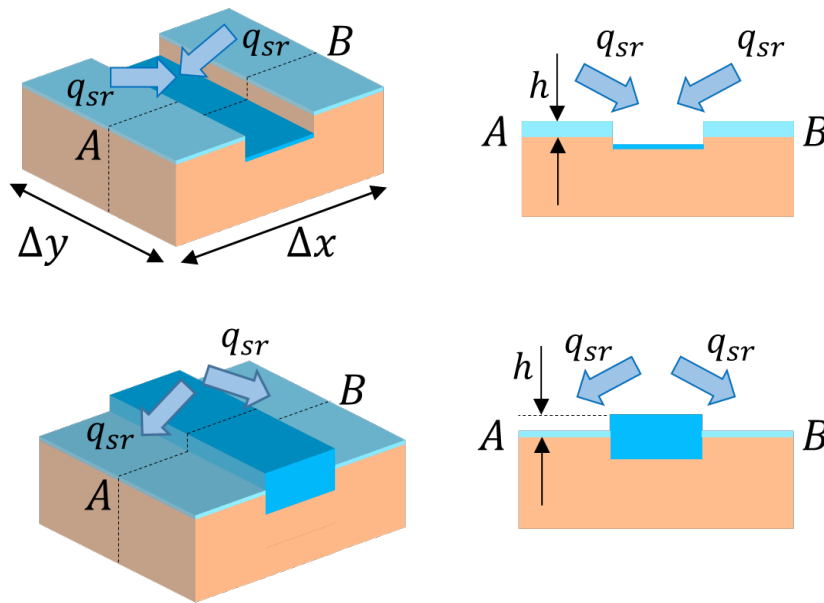


図4 斜面から河道への流量（上：河道の水位がグリッドセルの水位よりも低い場合、  
下：河道の水位がグリッドセルの水位よりも高い場合）

### 2.3.3. 河道における洪水流の支配方程式

河道における洪水流の質量保存則は次式で表される。

$$\frac{\partial h}{\partial t} + \frac{1}{B} \frac{\partial uhB}{\partial x} = \frac{2q_{sr}L}{BL} \quad (12)$$

ここに、 $B$ は川幅、 $L$ は流下方向の長さである。

洪水流の運動方程式には、 Manning 則を適用しており、次式で表される。

$$u = \frac{1}{n} \sqrt{\frac{\partial H}{\partial x}} h^{2/3} \quad (13)$$

ここに、 $n$ は Manning の粗度係数である。

### 2.3.4. 離散化

前述の支配方程式を図 5 に示す変数配置によって風上差分により離散化する。図の  $s$  (●)、 $u$  (▷)、 $v$  (△) は、それぞれスカラー、 $x$ 方向のベクトル、 $y$ 方向のベクトルを示す。 $s$  (●) は体積含水率 $p_w$ 、浸透流水深 $h_s$ 、表面流水深 $h$ 、最急勾配 $I$ 、最急勾配方向の浸透流 $q_s$ および表面流 $q$ を示す。 $u$  (▷) は、浸透流および浸透流フラックスの $x$ 成分である $q_{sx}$ 、 $q_x$ を示す。 $v$  (△) は、浸透流および表面流フラックスの $y$ 成分である $q_{sy}$ 、 $q_y$ を示す。時刻および位置に関する添え字を $(t)$ 、 $(i, j)$ とそれらの増分を $\Delta$ を用いて表すが、添え字に記載のないものは、時刻 $(t)$ 、位置 $(i, j)$ におけるものとする。 $i$ および $j$ の範囲は、スカラー型の変数では、それぞれ $i = 1$ から $i = iend$ および $j = 1$ から $j = jend$ 、ベクトル型の変数では、それぞれ $i = 1$ から $i = iend + 1$ および $j = 1$ から $j = jend + 1$ とする。境界条件はディリクレ境界条件（第 1 種境界条件）としている。

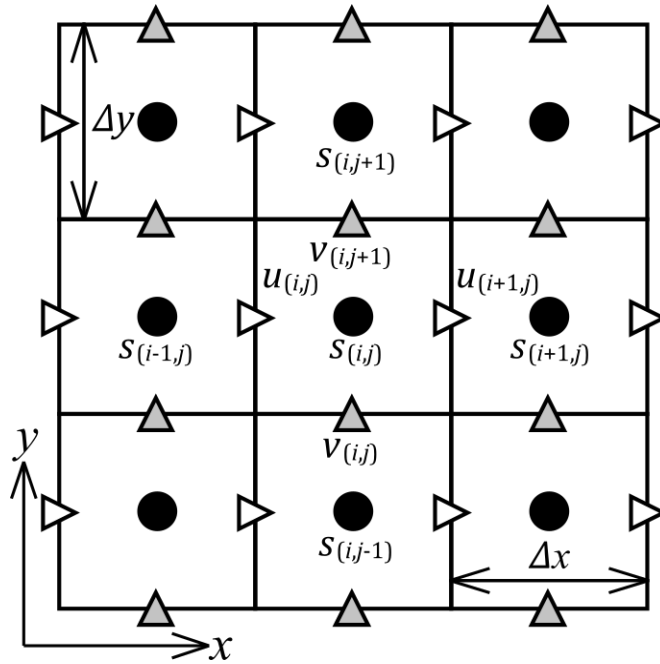


図 5 時刻 $t$ における変数配置図

#### 2.3.4.1. 質量保存則および運動量保存則の離散化

表土層の質量保存則を離散化すると、次式で表される。

$$\frac{p_w^{(t+\Delta t)} D - p_w^{(t)} D}{\Delta t} = r_{(t)} - e_{(t)} \quad (14)$$

浸透流の質量保存則を離散化すると、次式で表される。

$$\frac{h_s^{(t+\Delta t)} - h_s^{(t)}}{\Delta t} + \frac{1}{\lambda - p_w} \left( \frac{q_{sx(i+1,j)} - q_{sx(i,j)}}{\Delta x} + \frac{q_{sy(i,j+1)} - q_{sy(i,j)}}{\Delta y} \right) = \frac{\cos\theta}{\lambda - p_w} (f_{s1} - f_{s2} - e) \quad (15)$$

ダルシー則を適用して評価している最急勾配方向の浸透流の $x$ 成分および $y$ 成分は、それぞれ次式で近似している。

$$q_{sx(i,j)} = -k_{(i-1,j)} h_{(i-1,j)} \frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x} \quad 0 \leq \frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x} \quad (16)$$

$$q_{sx(i,j)} = -k_{(i,j)} h_{(i,j)} \frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x} \quad \frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x} < 0$$

$$q_{sy(i,j)} = -k_{(i,j-1)} h_{(i,j-1)} \frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y} \quad 0 \leq \frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y} \quad (17)$$

$$q_{sy(i,j)} = -k_{(i,j)} h_{(i,j)} \frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y} \quad \frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y} < 0$$

表面流の質量保存則を離散化すると次式で表される。

$$\frac{h^{(t+\Delta t)} - h^{(t)}}{\Delta t} + \left( \frac{q_x(i+1,j) - q_x(i,j)}{\Delta x} + \frac{q_y(i,j+1) - q_y(i,j)}{\Delta y} \right) = (r - f_{s2} - e) \cos \theta \quad (18)$$

Manning 則を適用して評価している最急勾配方向の表面流の $x$ 成分および $y$ 成分は、それぞれ次式で近似している。

$$q_x(i,j) = \frac{-1}{N_{(i-1,j)}} \sqrt{\frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x}} h_{(i-1,j)}^{5/3} \quad 0 \leq \frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x} \quad (19)$$

$$q_x(i,j) = \frac{1}{N_{(i,j)}} \sqrt{\left| \frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x} \right|} h_{(i,j)}^{5/3} \quad \frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x} < 0$$

$$q_y(i,j) = \frac{-1}{N_{(i,j-1)}} \sqrt{\frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y}} h_{(i,j-1)}^{5/3} \quad 0 \leq \frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y} \quad (20)$$

$$q_y(i,j) = \frac{1}{N_{(i,j)}} \sqrt{\left| \frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y} \right|} h_{(i,j)}^{5/3} \quad \frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y} < 0$$

#### 2.3.4.2. 境界条件

計算開始時刻 $t = 0$ における各変数の値を初期条件や初期値という。また、 $i = 1, i = iend, j = 1, j = jend$ における各変数の値を境界条件や境界値という。境界条件にディリクレ境界条件を適用している。浸透流の境界値は次式で表される。

$$q_{sx(1,j)} = -k_{(1,j)} h_{(1,j)} \frac{H_{(2,j)} - H_{(1,j)}}{\Delta x} \quad (21)$$

$$q_{sx(iend+1,j)} = -k_{(iend,j)} h_{(iend,j)} \frac{H_{(iend,j)} - H_{(iend-1,j)}}{\Delta x} \quad (22)$$

$$q_{sy(i,1)} = -k_{(i,1)}h_{(i,1)}\frac{H_{(i,2)} - H_{(i,1)}}{\Delta y} \quad (23)$$

$$q_{sy(i,jend+1)} = -k_{(i,jend)}h_{(i,jend)}\frac{H_{(i,jend)} - H_{(i,jend-1)}}{\Delta y} \quad (24)$$

表面流については式の変数が変わるだけで、採用する変数の空間関係は同様である。また、河道における洪水流の計算方法については、採用する変数の空間関係は土石流流出モデルと同様のため、後述する。

## 2.4. 土石流流出モデル

土石流流出モデルは、1次元で表現された河道において、斜面からの浸透流および表面流を境界条件とし、土石流の発生および侵食・堆積をともなう流下過程を計算する。土石流の発生は、河床堆積物に無限長斜面の安定解析<sup>17)</sup>を適用し、斜面からの流入量により不安定となった場合に、その地点の堆積物の全層が流動化するものとする。流動化した堆積物には、土石流の支配方程式<sup>10),11),18),19)</sup>を適用している。土石流の形成と規模の決定機構に重要な役割を果たす、微細砂の流体相への遷移現象<sup>20),16)</sup>を考慮するため、質量保存則は、固相として流れを支配する粗粒砂と、水とともに流体相として挙動する微細砂に分けて記述されている。

河床堆積物は様々な粒径の土粒子から構成される。ここでは、それらの土粒子を定数と考えられる閾値以上のものを粗粒砂、閾値未満のものを微細砂として扱う。図6に堆積物の粗粒土砂と微細土砂の静止、流動化、および堆積時における挙動を示す。粗粒砂および微細砂の含有率をそれぞれ、 $p_c$ および $p_f$  ( $p_c + p_f = 1$ )、堆積物における体積濃度を $c_*$ とすると、粗粒砂および微細砂はそれぞれ、 $p_c c_*$ および $p_f c_*$ で表される。堆積物が水で飽和され流動化すると、微細砂は流体相へ遷移して水とともに間隙流体を形成し、粗粒砂のみが固相を形成するとする。この時、粗粒砂の体積濃度 $c_c$ は $c_c = p_c c_*$ 、流体相における微細砂の体積濃度 $c_f$ は $c_f = p_f c_*/(1 - c_c)$ で表される。堆積時には、粗粒砂により形成された堆積物の空隙に水と微細砂で構成される間隙流体が取り込まれるものとする。このとき、微細砂を空隙に取り込んだ粗粒砂と微細砂の合計の土砂体積濃度 $c_{*D}$ は、 $c_{*D} = c_* + (1 - c_*)c_f$ で表される。

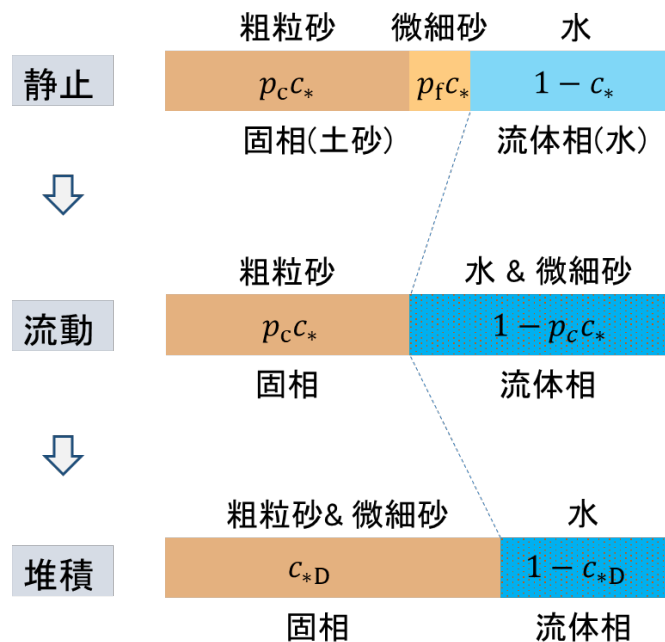


図6 堆積物の静止、流動化および堆積時の挙動<sup>17)</sup>

### 2.4.1. 河床堆積物の安定条件

表面流および浸透流が形成されている場において、無限長斜面の釣合条件を用いて崩壊の発生・非発生を判定する。この時、斜面に沿った重力成分 $G$ は、表土層、浸透流および表面流に作用する重力の流れ方向成分から構成され、抵抗力 $R$ はクーロン摩擦および粘着力から構成されるものとするれば、これらは次式で表される<sup>17)</sup>。

$$G = \rho g D \sin \theta \left\{ \left( \frac{\sigma}{\rho} \right) c_* + \left( 1 - \frac{h_s}{D} \right) p_w + \left( \frac{h_s}{D} \right) (1 - c_*) + \frac{h}{D} \right\} \quad (25)$$

$$R = \rho g D \cos \theta \left\{ \left( \frac{\sigma}{\rho} \right) c_* + \left( 1 - \frac{h_s}{D} \right) p_w - \left( \frac{h_s}{D} \right) c_* \right\} \tan \phi + c \quad (26)$$

ここに、 $\sigma$ は土粒子の質量密度、 $g$ は重力加速度、 $\rho$ は水の質量密度、 $c_*$ は表土層の体積土砂濃度、 $D$ は土層厚、 $\phi$ は内部摩擦角、 $c$ は表土層とその下層の境界における粘着力である。上式には浸透流による浮力の発生に伴い有効応力が減少することが考慮されている。崩壊の発生限界条件は $G = R$ より次式のようになる<sup>17)</sup>。

$$\tan \theta_c = \frac{\left( \frac{\sigma}{\rho} - \frac{h_s}{D} \right) c_* + \left( 1 - \frac{h_s}{D} \right) p_w + c'}{\left( \frac{\sigma}{\rho} - \frac{h_s}{D} \right) c_* + \frac{(h_s + h)}{D} + \left( 1 - \frac{h_s}{D} \right) p_w} \tan \phi \quad (27)$$

ここに、 $\theta_c$ は崩壊発生の限界勾配で、 $c'$ は粘着力の無次元量で、 $c' = c / (\rho g D \cos \theta \tan \phi)$ である。 $\theta_c < \theta$ の領域において崩壊が発生すると判定される。崩壊発生時には、堆積物の全層が流動化するものとする。

河床堆積物が非粘着性で構成され、水で飽和されている場合には、次式のようになる。

$$\tan \theta_c = \frac{(\sigma / \rho - 1) c_*}{(\sigma / \rho - 1) c_* + (1 + h / D)} \tan \phi \quad (28)$$

### 2.4.2. 土石流の支配方程式

#### 2.4.2.1. 質量保存則

解析モデルでは、微細砂の流体相への遷移後の間隙流体の質量密度を与えるため、流体相へ遷移した微細砂を質量で設定する必要がある。土石流全体、粗粒砂、および微細砂についての質量保存則<sup>18),19)</sup>を水深平均1次元で表すと、次式で表される。

侵食過程：

$$\frac{\partial h}{\partial t} + \frac{1}{B} \frac{\partial u h B}{\partial x} = \frac{E}{c_*} + f_{in} - f_{out} \quad (29)$$

$$\frac{\partial c_c h}{\partial t} + \frac{1}{B} \frac{\partial \gamma c_c u h B}{\partial x} = p_c E - c_c f_{out} \quad (30)$$

$$\frac{\partial c_f (1 - c_c) h}{\partial t} + \frac{1}{B} \frac{\partial c_f (1 - c_c) u h B}{\partial x} = p_f E - (1 - c_c) c_f f_{out} \quad (31)$$

堆積過程：

$$\frac{\partial h}{\partial t} + \frac{1}{B} \frac{\partial uhB}{\partial x} = \frac{E}{c_*} + f_{in} - f_{out} \quad (32)$$

$$\frac{\partial c_c h}{\partial t} + \frac{1}{B} \frac{\partial \gamma c_c uhB}{\partial x} = E - c_c f_{out} \quad (33)$$

$$\frac{\partial c_f (1 - c_c) h}{\partial t} + \frac{1}{B} \frac{\partial c_f (1 - c_c) uhB}{\partial x} = (1/c_* - 1) c_f E - (1 - c_c) c_f f_{out} \quad (34)$$

ここに、 $h$ は流動深、 $t$ は時間、 $B$ は川幅、 $x$ は河床に沿う軸、 $u$ は $x$ 軸方向の水深平均の流速、 $E$ は侵食・堆積速度、 $c_*$ は河床堆積物の静止体積濃度、 $f_{in}$ は流域斜面から河道への量、 $f_{out}$ は河道からの溢流量（後述の氾濫モデルの境界条件となる）、 $c_c$ は粗粒砂の水深平均の体積濃度、 $\gamma$ は土砂輸送補正係数、 $c_f$ は流体相における微細砂の体積濃度、 $c_{*D}$ は堆積過程における粗粒砂静止体積濃度である。侵食・堆積速度は、次式で表される<sup>11)</sup>。

$$\frac{E}{u} = c_* \tan(\theta - \theta_e) \quad (35)$$

ここに、 $\theta$ は河床勾配、 $\theta_e$ は土石流の平衡勾配であり、次式で表される<sup>11)</sup>。

$$\tan \theta_e = \frac{(\sigma/\rho - 1)c_c}{(\sigma/\rho - 1)c_c + 1} \tan \phi \quad (36)$$

ここに、 $\sigma$ は砂礫の質量密度、 $\rho$ は間隙流体の質量密度、および $\phi$ は砂礫の内部摩擦角である。平衡勾配の式における間隙流体の質量密度 $\rho$ は、流体相における微細砂濃度 $c_f$ を用いて次式で表される。

$$\rho = (\sigma - \rho_w)c_f + \rho_w \quad (37)$$

ここに、 $\rho_w$ は水の質量密度である。これにより、微細砂の流体相への遷移により間隙流体密度が増加することが表されている。上の2つの式において、微細砂の含有率 $p_f$ が増加すると、間隙流体密度 $\rho$ の増加と粗粒砂濃度 $c_c$ の減少により、平衡勾配 $\theta_e$ が小さくなることが表現されている。

河床堆積物の質量保存則は、次式で表される。

$$\frac{\partial z}{\partial t} = -\frac{E}{c_* \cos \theta} \quad (38)$$

ここに、 $z$ は河床の標高である。

#### 2.4.2.2. 運動量保存則

土石流の運動量保存則は、水深平均1次元で表すと次のように記述される。

$$\frac{\partial uh}{\partial t} + \frac{1}{B} \frac{\partial \beta uuhB}{\partial x} = -gh \frac{\partial H}{\partial x} - \frac{\tau_b}{\rho_m} \quad (39)$$

ここに、 $\beta$ は運動量補正係数、 $H$ は土石流の流れの表面の位置で $H = z + h \cos \theta$ で表され、 $\rho_m$ は土石流の質量密度、 $\tau_b$ は河床せん断力<sup>10)</sup>であり、次式で表される。

$$\tau_b = \tau_y + \rho f_b u^2 \quad (40)$$

ここに、 $\tau_y$ はクーロン型の降伏応力、 $f_b$ は流動抵抗係数である。これらは、それぞれ次式で表される。

$$\tau_y = \left(\frac{c_c}{c_*}\right)^{1/5} (\sigma - \rho)c_c g h \cos \theta \tan \phi \quad (41)$$

$$f_b = \frac{25}{4}(f_d + f_f) \left(\frac{h}{d}\right)^{-2} \quad (42)$$

$$f_d = k_d \left(\frac{\sigma}{\rho}\right) (1 - e^2)c_c^{1/3} \quad (43)$$

$$f_f = k_f(1 - c_c)^{5/3}c_c^{-2/3} \quad (44)$$

ここに、 $d$ は粗粒砂の代表粒径、 $k_f$ 、 $k_d$ は定数で、それぞれ、 $k_f = 0.16^{10)}$ 、 $k_d = 0.0828^{10)}$ 、 $e$ は砂礫同士における反発係数である。勾配が小さい領域や粒径が小さい場合に粒子流動層の上部に形成される乱流層の影響が卓越される場合を想定し、 $f_b$ に関する実測値<sup>27)</sup>を参考にして、 $f_b$ を次式<sup>28)</sup>によって評価し、式(42)による $f_b$ の値よりも大きい場合は、この値を採用する。

$$f_b = \left(A_r - \frac{1}{\kappa} + \frac{1}{\kappa} \ln \frac{h}{\kappa_s}\right)^{-2} \quad (45)$$

ここで、 $A_r$ は定数、 $\kappa$ はカルマン定数、および $\kappa_s$ は相当粗度で、当面 $A_r = 8.5^{28)}$ 、 $\kappa = 0.4^{28)}$ 、 $\kappa_s = d^{19)}$ を採用する。

#### 2.4.3. 土石流の溢水量の算出方法

土石流の流下過程において、土石流の表面標高が地盤標高を上回った地点では、土石流の水深に越流公式<sup>25)</sup>を適用して土石流の溢流量を算出し、これを土石流氾濫モデルに境界条件として与えている。本解析モデルでは、氾濫発生地点は、土石流が河道を流下していく過程で、土石流の表面標高と流路側岸の地盤標高によって定まる。図7は、河道から氾濫原に土石流が溢流する模式図である。土石流の溢流時における単位幅の溢流量は、便宜的に次式<sup>25)</sup>で評価している。

$$q_{sr} = \mu h \sqrt{2gh} \quad (46)$$

ここに、 $\mu$ は定数で $\mu = 0.35^{25)}$ 、 $g$ は重力加速度である。なお、一度溢流した土石流が氾濫原を流動し、河道に再び流入することは考慮していない。



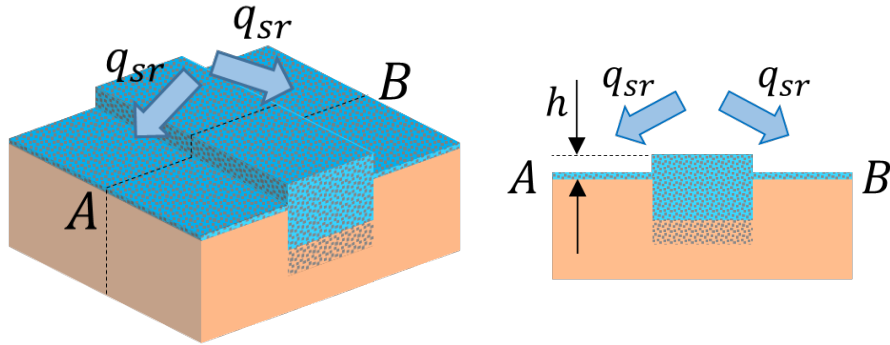


図 7 土石流の溢流

#### 2.4.4. 離散化

前述の支配方程式を図 8 に示す変数配置によって風上差分により離散化する。図の  $s$ 、 $u$  は、それぞれスカラー、 $x$ 方向のベクトルを示す。 $s$ は水深 $h$ 、粗粒砂濃度 $c_c$ 、微細砂濃度 $c_f$ を示し、 $u$ は、流速を示す。時刻および位置に関する添え字を $(t)$ 、 $(i)$ とそれらの増分を $\Delta$ を用いて表すが、添え字に記載のないものは、時刻 $(t)$ 、位置 $(i)$ におけるものとする。

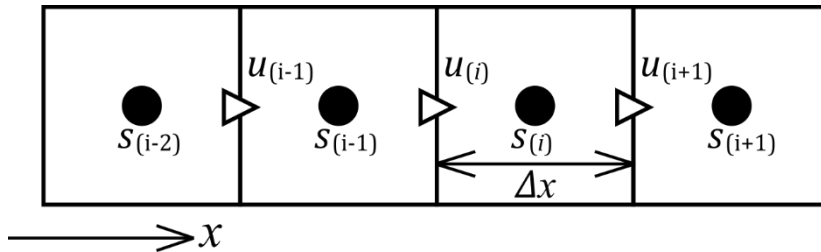


図 8 変数配置図

##### 2.4.4.1. 質量保存則

土石流全体、粗粒砂、および微細砂の質量保存則の侵食過程を離散化すると、次のように表される。

土石流全体の質量保存則(29)を離散化すると次式で表される。

$$\frac{h^{(t+\Delta t)} - h^{(t)}}{\Delta t} + \frac{1}{B} \frac{M_{(i+1)} - M_{(i)}}{\Delta x} = \frac{E}{c_*} + f_{in} - f_{out} \quad (47)$$

ここに、 $M$ は $u$ の位置における水・土砂の全流量で、次式で表される。

$$\begin{aligned} M_{(i)} &= u_{(i)} h_{(i-1)} B_{(i-1)} & 0 \leq u_{(i)} \\ M_{(i)} &= u_{(i)} h_{(i)} B_{(i)} & u_{(i)} < 0 \end{aligned} \quad (48)$$

$$\begin{aligned} M_{(i+1)} &= u_{(i+1)} h_{(i)} B_{(i)} & 0 \leq u_{(i+1)} \\ M_{(i+1)} &= u_{(i+1)} h_{(i+1)} B_{(i+1)} & u_{(i+1)} < 0 \end{aligned} \quad (49)$$

粗粒砂の質量保存則(30)を離散化すると次式で表される。

$$\frac{c_c^{(t+\Delta t)}h^{(t+\Delta t)} - c_c^{(t)}h^{(t)}}{\Delta t} + \frac{1}{B} \frac{M_{(i+1)} - M_{(i)}}{\Delta x} = p_c E - c_c f_{out} \quad (50)$$

ここに、 $M$ は $u$ の位置で定義される粗粒砂の全流量で、次式で表される。

$$M_{(i)} = \gamma_{(i-1)} c_{c(i-1)} u_{(i)} h_{(i-1)} B_{(i-1)} \quad 0 \leq u_{(i)} \quad (51)$$

$$M_{(i)} = \gamma_{(i)} c_{c(i)} u_{(i)} h_{(i)} B_{(i)} \quad u_{(i)} < 0$$

$$M_{(i+1)} = \gamma_{(i)} c_{c(i)} u_{(i+1)} h_{(i)} B_{(i)} \quad 0 \leq u_{(i+1)} \quad (52)$$

$$M_{(i+1)} = \gamma_{(i+1)} c_{c(i+1)} u_{(i+1)} h_{(i+1)} B_{(i+1)} \quad u_{(i+1)} < 0$$

$h^{(t+\Delta t)}$ を既知とする必要が生ずるが、プログラムの計算順序で先に計算されているため、疑似的に $t + \Delta t$ のものとなししている。

微細砂の質量保存則(31)を離散化すると次式で表される。

$$\frac{c_f^{(t+\Delta t)}(1 - c_c^{(t+\Delta t)})h^{(t+\Delta t)} - c_f^{(t)}(1 - c_c^{(t)})h^{(t)}}{\Delta t} + \frac{1}{B} \frac{M_{(i+1)} - M_{(i)}}{\Delta x} = p_f E - (1 - c_c) c_f f_{out} \quad (53)$$

ここに、 $M$ は $u$ の位置で定義される微細砂の全流量で、次式で表される。

$$M_{(i)} = c_{f(i-1)}(1 - c_{c(i-1)})u_{(i)}h_{(i-1)}B_{(i-1)} \quad 0 \leq u_{(i)} \quad (54)$$

$$M_{(i)} = c_{f(i)}(1 - c_{c(i)})u_{(i)}h_{(i)}B_{(i)} \quad u_{(i)} < 0$$

$$M_{(i+1)} = c_{f(i)}(1 - c_{c(i)})u_{(i+1)}h_{(i)}B_{(i)} \quad 0 \leq u_{(i+1)} \quad (55)$$

$$M_{(i+1)} = c_{f(i+1)}(1 - c_{c(i+1)})u_{(i+1)}h_{(i+1)}B_{(i+1)} \quad u_{(i+1)} < 0$$

以上、土石流全体、粗粒砂、および微細砂の質量保存則の堆積過程における離散化においても、左辺は同様である。

河床堆積物の質量保存則を離散化すると、次式で表される。

$$\frac{z^{(t+\Delta t)} - z^{(t)}}{\Delta t} = - \frac{E}{c_* \cos \theta} \quad (56)$$

#### 2.4.4.2. 合流点の計算方法

合流は、式(48)、(51)、(54)における $M_{(i)}$ において考慮する。図 9 に本川支川の合流点における変数配置を示す。本解析では、合流点において、より大きい集水面積を持つ河道を本川、それ以外を支川としている。ここに、本川および支川の添字を、それぞれ $m$ 、 $t$ としている。式(48)を例にすると、 $M_{(i)}$ は次式で表される。

$$\begin{aligned} M_{(i)} &= u_{(i_m)} h_{(i_m-1)} B_{(i_m-1)} + u_{(i_t)} h_{(i_t-1)} B_{(i_t-1)} & 0 \leq u_{(i_m)} & \quad 0 \leq u_{(i_t)} \\ M_{(i)} &= u_{(i_m)} h_{(i_m-1)} B_{(i_m-1)} + u_{(i_t)} h_{(i_m)} B_{(i_m)} & 0 \leq u_{(i_m)} & \quad u_{(i_t)} < 0 \\ M_{(i)} &= u_{(i_m)} h_{(i_m)} B_{(i_m)} + u_{(i_t)} h_{(i_t-1)} B_{(i_t-1)} & u_{(i_m)} < 0 & \quad 0 \leq u_{(i_t)} \\ M_{(i)} &= u_{(i_m)} h_{(i_m)} B_{(i_m)} + u_{(i_t)} h_{(i_m)} B_{(i_m)} & u_{(i_m)} < 0 & \quad u_{(i_t)} < 0 \end{aligned} \quad (57)$$

これは、粗粒砂および微細砂における連続条件についても同様である。また、逆流する場合には、本川のみ逆流するものとしている。

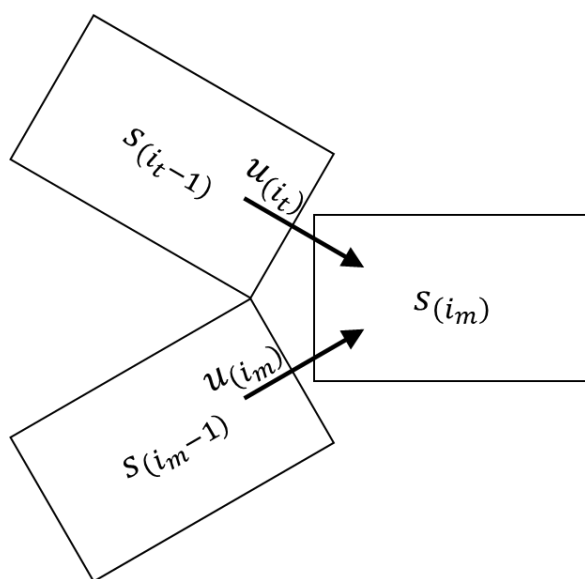


図 9 合流点の変数配置図

#### 2.4.4.3. 運動量保存則

運動量保存則(39)の左辺第二項および右辺第一項をそれぞれ $X$ 、 $Y$ として、離散化すると次式で表される。

$0 \leq u_{(i)}$ の時、次式で表される。

$$\frac{u_{(i)}^{(t+\Delta t)} h_{(i-1)}^{(t+\Delta t)} - u_{(i)}^{(t)} h_{(i-1)}^{(t)}}{\Delta t} + X = -Y - \frac{\tau_{y(i-1)} + \rho_{(i-1)} f_{b(i-1)} u_{(i)}^{(t+\Delta t)} |u_{(i)}^{(t)}|}{\rho_{m(i-1)}} \quad (58)$$

ここに、右辺の第二項には、Vasiliev 不安定回避<sup>29)</sup>のため、 $u_{(i)}^{(t+\Delta t)}$ を用いている。 $h_{(i-1)}^{(t+\Delta t)}$ を既知とする必要が生ずるが、プログラムの計算順序で先に計算されているため、疑似的に $t + \Delta t$ のものとみなしている。

$X$ は次式で表される。

$$X = u_{(i)} \frac{1}{B_{(i-1)}} \frac{M_{(i)} - M_{(i-1)}}{\Delta x} \quad (59)$$

ここに、 $M$ は $u$ の位置で定義される水および土砂の全流量で、次式で表される。

$$M_{(i)} = \beta_{(i-1)} u_{(i)} h_{(i-1)} B_{(i-1)} \quad 0 \leq u_{(i)} \quad (60)$$

$$M_{(i-1)} = \beta_{(i-2)} u_{(i-1)} h_{(i-2)} B_{(i-2)} \quad 0 \leq u_{(i-1)} \quad (61)$$

$$M_{(i-1)} = \beta_{(i-1)} u_{(i-1)} h_{(i-1)} B_{(i-1)} \quad u_{(i-1)} < 0$$

$Y$ は次式で表される。

$$Y = g h_{(i-1)} \frac{H_{(i)} - H_{(i-1)}}{\Delta x} \quad (62)$$

$u_{(i)} < 0$ の時は次式で表される。

$$\frac{u_{(i)}^{(t+\Delta t)} h_{(i)}^{(t+\Delta t)} - u_{(i)}^{(t)} h_{(i)}^{(t)}}{\Delta t} + X = -Y - \frac{\tau_{y(i)} + \rho_{(i)} f_b(i) u_{(i)}^{(t+\Delta t)} |u_{(i)}^{(t)}|}{\rho_m(i)} \quad (63)$$

$X$ は次式で表される。

$$X = u_{(i)} \frac{1}{B_{(i)}} \frac{M_{(i+1)} - M_{(i)}}{\Delta x} \quad (64)$$

ここに、 $M$ は $u$ の位置で定義される水および土砂の全流量で、次式で表される。

$$M_{(i+1)} = \beta_{(i)} u_{(i+1)} h_{(i)} B_{(i)} \quad 0 \leq u_{(i+1)} \quad (65)$$

$$M_{(i+1)} = \beta_{(i+1)} u_{(i+1)} h_{(i+1)} B_{(i+1)} \quad u_{(i+1)} < 0$$

$$M_{(i)} = \beta_{(i)} u_{(i)} h_{(i)} B_{(i)} \quad u_{(i)} < 0 \quad (66)$$

$Y$ は次式で表される。

$$Y = g h_{(i)} \frac{H_{(i)} - H_{(i-1)}}{\Delta x} \quad (67)$$

合流点においては、 $0 \leq u_{(i)}$ かつ $0 \leq u_{(i-1)}$ の場合のみ、式(57)と同様に計算を行い、それ以外の場合では、本川にのみ逆流する。

## 2.5. 土石流氾濫モデル

土石流氾濫モデルは、河道から溢流した土石流の氾濫過程を計算する。土石流の支配方程式は、平面2次元座標における表現であることを除き、土石流流出モデルの支配方程式と同様である。

### 2.5.1. 土石流の支配方程式

土石流全体、粗粒砂、微細砂、および河床堆積物の質量保存則<sup>18),19)</sup>において、侵食過程は次式で表される。

$$\frac{\partial h}{\partial t} + \frac{\partial uh}{\partial x} + \frac{\partial vh}{\partial y} = \frac{E}{c_*} + f_{df} \quad (68)$$

$$\frac{\partial c_c h}{\partial t} + \frac{\partial \gamma c_c uh}{\partial x} + \frac{\partial c_c vh}{\partial y} = p_c E + f_{cc} \quad (69)$$

$$\frac{\partial c_f(1-c_c)h}{\partial t} + \frac{\partial c_f(1-c_c)uh}{\partial x} + \frac{\partial c_f(1-c_c)vh}{\partial y} = p_f E + f_{cf} \quad (70)$$

堆積過程は次式で表される。

$$\frac{\partial h}{\partial t} + \frac{\partial uh}{\partial x} + \frac{\partial vh}{\partial y} = \frac{E}{c_{*D}} + f_{df} \quad (71)$$

$$\frac{\partial c_c h}{\partial t} + \frac{\partial \gamma c_c uh}{\partial x} + \frac{\partial c_c vh}{\partial y} = E + f_{cc} \quad (72)$$

$$\frac{\partial c_f(1-c_c)h}{\partial t} + \frac{\partial c_f(1-c_c)uh}{\partial x} + \frac{\partial c_f(1-c_c)vh}{\partial y} = (1/c_* - 1)c_f E + f_{cf} \quad (73)$$

ここに、 $h$ は流動深、 $t$ は時間、 $x$ 、 $y$ は直交直線座標で、 $x-y$ 平面は水深平均流速ベクトルが作る面に一致している、 $u$ は $x$ 軸方向の水深平均の流速、 $v$ は $y$ 軸方向の水深平均の流速、 $E$ は侵食・堆積速度、 $c_*$ は河床堆積物の静止体積濃度、 $f_{df}$ は河道から氾濫原への土石流の溢流、 $c_c$ は粗粒砂の水深平均の体積濃度、 $\gamma$ は土砂輸送補正係数、 $f_{cc}$ は河道から氾濫原への土石流における粗粒砂の溢流、 $c_f$ は流体相における微細砂の体積濃度、 $c_{*D}$ は堆積過程における粗粒砂静止体積濃度、 $f_{cf}$ は河道から氾濫原への土石流における微細砂の溢流である。侵食・堆積速度は、次式で表される<sup>11)</sup>。

$$\frac{E}{\sqrt{u^2 + v^2}} = c_* \tan(\theta - \theta_e) \quad (74)$$

ここに、 $\theta$ は土石流の流速ベクトルと水平面となす角、 $\theta_e$ は平衡勾配で、それぞれ次式で表される。

$$\tan \theta = \left( -\frac{\partial z}{\partial x} u - \frac{\partial z}{\partial y} v \right) \frac{1}{\sqrt{u^2 + v^2}} \quad (75)$$

$$\tan \theta_e = \frac{(\sigma/\rho - 1)c_c}{(\sigma/\rho - 1)c_c + 1} \tan \phi \quad (76)$$

ここに、 $\sigma$ は砂礫の質量密度、 $\rho$ は間隙流体の質量密度で次式で表される。

$$\rho = (\sigma - \rho_w)c_f + \rho_w \quad (77)$$

運動量保存則は次式で表される。

$$\frac{\partial uh}{\partial t} + \frac{\partial \beta uuh}{\partial x} + \frac{\partial \beta vuh}{\partial y} = -gh \frac{\partial H}{\partial x} - \frac{\tau_{bx}}{\rho_m} \quad (78)$$

$$\frac{\partial vh}{\partial t} + \frac{\partial \beta uvh}{\partial x} + \frac{\partial \beta vvh}{\partial y} = -gh \frac{\partial H}{\partial y} - \frac{\tau_{by}}{\rho_m} \quad (79)$$

ここに、 $\beta$ は運動量補正係数、 $H$ は土石流の流れの表面位置、 $\tau_{bx}$ 、 $\tau_{by}$ はそれぞれ河床せん断力 $\tau_b$ <sup>10)</sup>の $x$ 、 $y$ 方向成分、 $\rho_m$ は土石流の質量密度であり、次式で表される。

$$\rho_m = (\sigma - \rho)c_c + \rho \quad (80)$$

河床せん断力 $\tau_b$ とその $x$ 、 $y$ 方向成分 $\tau_{bx}$ 、 $\tau_{by}$ はそれぞれ次式で表される。

$$\tau_{bx} = \frac{\tau_b u}{\sqrt{u^2 + v^2}} \quad (81)$$

$$\tau_{by} = \frac{\tau_b v}{\sqrt{u^2 + v^2}} \quad (82)$$

$$\tau_b = \tau_y + \rho f_b (u^2 + v^2) \quad (83)$$

ここに、 $\tau_y$ はクーロン型の降伏応力、 $f_b$ は流動抵抗係数であって、それぞれ次式で与えられる。

$$\tau_y = \left(\frac{c_c}{c_*}\right)^{1/5} (\sigma - \rho)c_c gh \cos \theta \tan \phi \quad (84)$$

$$f_b = \frac{25}{4} (f_a + f_f) \left(\frac{h}{d}\right)^{-2} \quad (85)$$

$$f_a = k_a (\sigma/\rho) (1 - e^2) c_c^{1/3} \quad (86)$$

$$f_f = k_f (1 - c_c)^{5/3} c_c^{-2/3} \quad (87)$$

ここに、 $d$ は粗粒砂の粒径、 $k_f = 0.16$ 、 $k_a = 0.0828$ 、 $e$ は砂礫同士における反発係数である。勾配が小さい領域や粒径が小さい場合に粒子流動層の上部に形成される乱流層の影響が卓越される場合を想定し $f_b$ に関する実測値<sup>27)</sup>を参考にして $f_b$ を次式<sup>28)</sup>によって評価し、式(85)による $f_b$ の値よりも大きい場合は、この値を採用する。

$$f_b = \left(A_r - \frac{1}{\kappa} + \frac{1}{\kappa} \ln \frac{h}{\kappa_s}\right)^{-2} \quad (88)$$

ここで、 $A_r$ は定数、 $\kappa$ はカルマン定数、および $\kappa_s$ は相当粗度で、当面 $A_r = 8.5$ <sup>28)</sup>、 $\kappa = 0.4$ <sup>28)</sup>、 $\kappa_s = d$ <sup>19)</sup>を採用する。

### 2.5.2. 離散化

前述の支配方程式を図 10 に示す変数配置によって風上差分により離散化する。図の $s$ 、 $u$ 、 $v$ は、それぞれスカラー、 $x$ 方向のベクトル、 $y$ 方向のベクトルを示す。 $s$ は水深 $h$ 、粗粒

砂濃度 $c_c$ 、土砂輸送補正係数 $\gamma$ 、微細砂濃度 $c_f$ を示し、 $u$ 、 $v$ は、流速を示す。時刻および位置に関する添え字を $(t)$ 、 $(i, j)$ とそれらの増分を $\Delta$ を用いて表すが、添え字に記載のないものは、時刻 $(t)$ 、位置 $(i, j)$ におけるものとする。

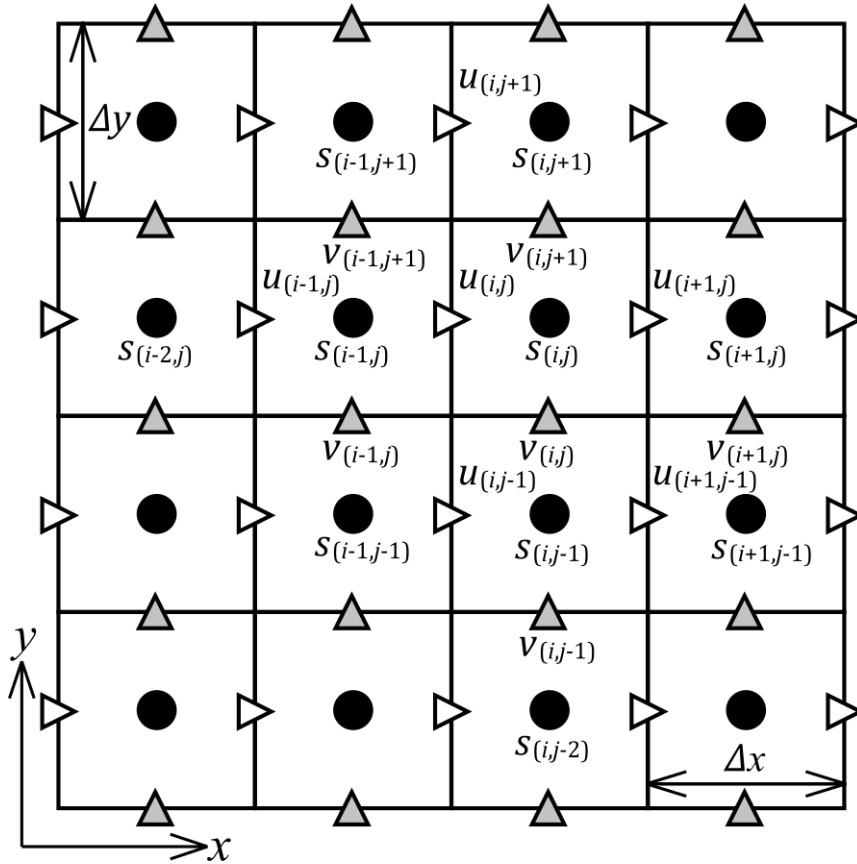


図 10 変数配置図

### 2.5.2.1. 質量保存則

土石流全体、粗粒砂、および微細砂の質量保存則の侵食過程を離散化すると、次のように表される。土石流全体の質量保存則(68)を離散化すると次式で表される。

$$\frac{h^{(t+\Delta t)} - h^{(t)}}{\Delta t} + \frac{M_{(i+1, j)} - M_{(i, j)}}{\Delta x} + \frac{N_{(i, j+1)} - N_{(i, j)}}{\Delta y} = \frac{E}{c_*} + f_{df} \quad (89)$$

ここに、 $M$ は $u$ の位置における水・土砂の全流量、 $N$ は $v$ の位置における水・土砂の全流量で次式で表される。

$$\begin{aligned} M_{(i, j)} &= u_{(i, j)} h_{(i-1, j)} & 0 \leq u_{(i, j)} \\ M_{(i, j)} &= u_{(i, j)} h_{(i, j)} & u_{(i, j)} < 0 \end{aligned} \quad (90)$$

$$\begin{aligned} M_{(i+1, j)} &= u_{(i+1, j)} h_{(i, j)} & 0 \leq u_{(i+1, j)} \\ M_{(i+1, j)} &= u_{(i+1, j)} h_{(i+1, j)} & u_{(i+1, j)} < 0 \end{aligned} \quad (91)$$

$$\begin{aligned} N_{(i, j)} &= v_{(i, j)} h_{(i, j-1)} & 0 \leq v_{(i, j)} \end{aligned} \quad (92)$$

$$\begin{aligned}
N_{(i,j)} &= v_{(i,j)} h_{(i,j)} & v_{(i,j)} &< 0 \\
N_{(i,j+1)} &= v_{(i,j+1)} h_{(i,j)} & 0 &\leq v_{(i,j+1)} \\
N_{(i,j+1)} &= v_{(i,j+1)} h_{(i,j+1)} & v_{(i,j+1)} &< 0
\end{aligned} \tag{93}$$

粗粒砂の質量保存則(69)を離散化すると次式で表される。

$$\frac{c_c^{(t+\Delta t)} h^{(t+\Delta t)} - c_c^{(t)} h^{(t)}}{\Delta t} + \frac{M_{(i+1,j)} - M_{(i,j)}}{\Delta x} + \frac{N_{(i,j+1)} - N_{(i,j)}}{\Delta y} = p_c E + f_{cc} \tag{94}$$

ここに、 $M$ は $u$ の位置における粗粒砂の全流量、 $N$ は $v$ の位置における粗粒砂の全流量で次式で表される。

$$M_{(i,j)} = c_c^{(i-1,j)} \gamma_{(i-1,j)} u_{(i,j)} h_{(i-1,j)} \quad 0 \leq u_{(i,j)} \tag{95}$$

$$M_{(i,j)} = c_c^{(i,j)} \gamma_{(i,j)} u_{(i,j)} h_{(i,j)} \quad u_{(i,j)} < 0$$

$$M_{(i+1,j)} = c_c^{(i,j)} \gamma_{(i,j)} u_{(i+1,j)} h_{(i,j)} \quad 0 \leq u_{(i+1,j)} \tag{96}$$

$$M_{(i+1,j)} = c_c^{(i+1,j)} \gamma_{(i+1,j)} u_{(i+1,j)} h_{(i+1,j)} \quad u_{(i+1,j)} < 0$$

$$N_{(i,j)} = c_c^{(i,j-1)} \gamma_{(i,j-1)} v_{(i,j)} h_{(i,j-1)} \quad 0 \leq v_{(i,j)} \tag{97}$$

$$N_{(i,j)} = c_c^{(i,j)} \gamma_{(i,j)} v_{(i,j)} h_{(i,j)} \quad v_{(i,j)} < 0$$

$$N_{(i,j+1)} = c_c^{(i,j)} \gamma_{(i,j)} v_{(i,j+1)} h_{(i,j)} \quad 0 \leq v_{(i,j+1)} \tag{98}$$

$$N_{(i,j+1)} = c_c^{(i,j+1)} \gamma_{(i,j+1)} v_{(i,j+1)} h_{(i,j+1)} \quad v_{(i,j+1)} < 0$$

$h^{(t+\Delta t)}$ を既知とする必要が生ずるが、プログラムの計算順序で先に計算されているため、これを疑似的に $t + \Delta t$ のものみなしている。

微細砂の質量保存則(70)を離散化すると次式で表される。

$$\begin{aligned}
&\frac{c_f^{(t+\Delta t)} (1 - c_c^{(t+\Delta t)}) h^{(t+\Delta t)} - c_f^{(t)} (1 - c_c^{(t)}) h^{(t)}}{\Delta t} \\
&+ \frac{M_{(i+1,j)} - M_{(i,j)}}{\Delta x} + \frac{N_{(i,j+1)} - N_{(i,j)}}{\Delta y} = p_f E + f_{cf}
\end{aligned} \tag{99}$$

ここに、 $M$ は $u$ の位置における微細砂の全流量、 $N$ は $v$ の位置における微細砂の全流量で次式で表される。

$$M_{(i,j)} = c_f^{(i-1,j)} (1 - c_c^{(i-1,j)}) u_{(i,j)} h_{(i-1,j)} \quad 0 \leq u_{(i,j)} \tag{100}$$

$$M_{(i,j)} = c_f^{(i,j)} (1 - c_c^{(i,j)}) u_{(i,j)} h_{(i,j)} \quad u_{(i,j)} < 0$$

$$M_{(i+1,j)} = c_f^{(i,j)} (1 - c_c^{(i,j)}) u_{(i+1,j)} h_{(i,j)} \quad 0 \leq u_{(i+1,j)} \tag{101}$$

$$M_{(i+1,j)} = c_f^{(i+1,j)} (1 - c_c^{(i+1,j)}) u_{(i+1,j)} h_{(i+1,j)} \quad u_{(i+1,j)} < 0$$

$$N_{(i,j)} = c_f^{(i,j-1)} (1 - c_c^{(i,j-1)}) v_{(i,j)} h_{(i,j-1)} \quad 0 \leq v_{(i,j)} \tag{102}$$

$$N_{(i,j)} = c_f^{(i,j)} (1 - c_c^{(i,j)}) v_{(i,j)} h_{(i,j)} \quad v_{(i,j)} < 0$$

$$N_{(i,j+1)} = c_f^{(i,j)} (1 - c_c^{(i,j)}) v_{(i,j+1)} h_{(i,j)} \quad 0 \leq v_{(i,j+1)} \tag{103}$$

$$N_{(i,j+1)} = c_f^{(i,j+1)} (1 - c_c^{(i,j+1)}) v_{(i,j+1)} h_{(i,j+1)} \quad v_{(i,j+1)} < 0$$

### 2.5.2.1. 運動量保存則

$x$ 方向における運動量保存則(39)の左辺第二項および第三項を $X$ 、右辺第一項 $Y$ として、離



散化すると次式で表される。

$$\frac{u^{(t+\Delta t)}\bar{h}^{(t+\Delta t)} - u^{(t)}\bar{h}^{(t)}}{\Delta t} = -X - Y - \frac{\bar{\tau}_b}{\bar{\rho}_m} \frac{u^{(t+\Delta t)} + u^{(t)}}{2\sqrt{u^{(t)^2} + \bar{v}^2}} \quad (104)$$

右辺第三項には、Vasiliev 不安定回避<sup>29)</sup>のため、 $u^{(t)}$ を $(u^{(t+\Delta t)} + u^{(t)})/2$ に置き換えている。 $h^{(t+\Delta t)}$ を既知とする必要が生ずるが、プログラムの計算順序で先に計算されているため、これを疑似的に $t + \Delta t$ のもののみなしている。ここに、 $\bar{h}$ 、 $\bar{\tau}_b$ 、 $\bar{\rho}_m$ 、 $\bar{v}$ はそれぞれ、 $h$ 、 $\tau_b$ 、 $\rho_m$ 、 $v$ の空間的な平均値であり次式で表される。

$$\bar{h} = (h_{(i-1,j)} + h_{(i,j)})/2 \quad (105)$$

$$\bar{\tau}_b = (\tau_{b(i-1,j)} + \tau_{b(i,j)})/2 \quad (106)$$

$$\bar{\rho}_m = (\rho_{m(i-1,j)} + \rho_{m(i,j)})/2 \quad (107)$$

$$\bar{v} = (v_{(i,j)} + v_{(i,j+1)} + v_{(i-1,j+1)} + v_{(i-1,j)})/4 \quad (108)$$

$X$ は次式で表される。

$$X = u \frac{M_{(i,j)} - M_{(i-1,j)}}{\Delta x} + \bar{v} \frac{M_{(i,j)} - M_{(i,j-1)}}{\Delta y} \quad 0 \leq u \quad 0 \leq \bar{v} \quad (109)$$

$$X = u \frac{M_{(i,j)} - M_{(i-1,j)}}{\Delta x} + \bar{v} \frac{M_{(i,j+1)} - M_{(i,j)}}{\Delta y} \quad 0 \leq u \quad \bar{v} < 0 \quad (110)$$

$$X = u \frac{M_{(i+1,j)} - M_{(i,j)}}{\Delta x} + \bar{v} \frac{M_{(i,j)} - M_{(i,j-1)}}{\Delta y} \quad u < 0 \quad 0 \leq \bar{v} \quad (111)$$

$$X = u \frac{M_{(i+1,j)} - M_{(i,j)}}{\Delta x} + \bar{v} \frac{M_{(i,j+1)} - M_{(i,j)}}{\Delta y} \quad u < 0 \quad \bar{v} < 0 \quad (112)$$

ここに、 $M$ は $u$ の位置における水・土砂の全流量で次式で表される。

$$M = \bar{\beta} u \bar{h} \quad (113)$$

ここに、 $\bar{\beta}$ は $\beta$ の空間的な平均値であり次式で表される。

$$\bar{\beta} = (\beta_{(i-1,j)} + \beta_{(i,j)})/2 \quad (114)$$

$Y$ は次式で表される。

$$Y = g \bar{h} \frac{H_{(i,j)} - H_{(i-1,j)}}{\Delta x} \quad (115)$$

$y$ 方向における運動量保存則(39)の左辺第二項および第三項を $X$ 、右辺第一項を $Y$ として、離散化すると次式で表される。

$$\frac{v^{(t+\Delta t)}\bar{h}^{(t+\Delta t)} - v^{(t)}\bar{h}^{(t)}}{\Delta t} = -X - Y - \frac{\bar{\tau}_b}{\bar{\rho}_m} \frac{v^{(t+\Delta t)} + v^{(t)}}{2\sqrt{v^{(t)^2} + \bar{u}^2}} \quad (116)$$

右辺第三項には、Vasiliev 不安定回避<sup>29)</sup>のため、 $v^{(t)}$ を $(v^{(t+\Delta t)} + v^{(t)})/2$ に置き換えている。ここに、 $\bar{h}$ 、 $\bar{\tau}_b$ 、 $\bar{\rho}_m$ 、 $\bar{v}$ はそれぞれ、 $h$ 、 $\tau_b$ 、 $\rho_m$ 、 $u$ の空間的な平均値であり次式で表される。

$$\bar{h} = (h_{(i,j-1)} + h_{(i,j)})/2 \quad (117)$$

$$\bar{\tau}_b = (\tau_{b(i,j-1)} + \tau_{b(i,j)})/2 \quad (118)$$

$$\bar{\rho}_m = (\rho_{m(i,j-1)} + \rho_{m(i,j)})/2 \quad (119)$$

$$\bar{u} = (u_{(i,j)} + u_{(i,j-1)} + u_{(i+1,j-1)} + u_{(i+1,j)})/4 \quad (120)$$

$X$ は次式で表される。

$$X = \bar{u} \frac{N_{(i,j)} - N_{(i-1,j)}}{\Delta x} + v \frac{N_{(i,j)} - N_{(i,j-1)}}{\Delta y} \quad 0 \leq \bar{u} \quad 0 \leq v \quad (121)$$

$$X = \bar{u} \frac{N_{(i,j)} - N_{(i-1,j)}}{\Delta x} + v \frac{N_{(i,j+1)} - N_{(i,j)}}{\Delta y} \quad 0 \leq \bar{u} \quad v < 0 \quad (122)$$

$$X = \bar{u} \frac{N_{(i+1,j)} - N_{(i,j)}}{\Delta x} + v \frac{N_{(i,j)} - N_{(i,j-1)}}{\Delta y} \quad \bar{u} < 0 \quad 0 \leq v \quad (123)$$

$$X = \bar{u} \frac{N_{(i+1,j)} - N_{(i,j)}}{\Delta x} + v \frac{N_{(i,j+1)} - N_{(i,j)}}{\Delta y} \quad \bar{u} < 0 \quad v < 0 \quad (124)$$

ここに、 $N$ は $v$ の位置における水・土砂の全流量で次式で表される。

$$N = \bar{\beta} v \bar{h} \quad (125)$$

ここに、 $\bar{\beta}$ は $\beta$ の空間的な平均値であり次式で表される。

$$\bar{\beta} = (\beta_{(i,j-1)} + \beta_{(i,j)})/2 \quad (126)$$

$Y$ は次式で表される。

$$Y = g \bar{h} \frac{H_{(i,j)} - H_{(i,j-1)}}{\Delta y} \quad (127)$$

### 3. モデルの適用事例

本モデルにおける降雨流出、土石流流出および土石流氾濫それぞれの解析モデルについて、実際に発生した事例に適用し、その再現性とモデルの設定値を検討する。検討は次の3つの事例について行う。

#### 1. 降雨流出

2017年九州北部豪雨における寺内ダム流域に適用する。寺内ダムのダム流入量データは、国土交通省水文水質データベース<sup>30)</sup>にて公開されている。ダム流入量を、ダムを流域下端とした場合の流出量と同様とみなし、ダム流入量を用いて降雨流出計算の検証を行う。2017年九州北部豪雨時における寺内ダムに降雨流出モデルを適用し、その結果とダム流入量のデータを比較して、降雨流出解析モデルの適用性を検討する。

#### 2. 土石流流出

桜島有村川において2014年8月に発生した土石流に適用する。桜島有村川では、有村川3号堰堤において、土石流の表面流速、表面形状、荷重が計測されている。このデータを用いて、土石流の水深平均流速および流動深の時間変化を算出し、計算結果を比較して、土石流流出解析モデルの適用性を検討する。

#### 3. 土石流氾濫

1999年九州北部豪雨時における阿蘇古恵川の土石流に適用する。1990年7月の豪雨により、阿蘇市一の宮町では古恵川から土石流が氾濫し、土砂災害が発生した。この災害においては、災害の発生時刻、土石流の痕跡、氾濫範囲や氾濫した土砂量に関する調査結果<sup>31)</sup>がある。これらの情報を用いて、土石流流出および土石流氾濫解析モデルの再現性を検討する。

### 3.1. 降雨流出

2017年九州北部豪雨における寺内ダム流域に適用する。寺内ダムのダム流入量データは、国土交通省水文水質データベース<sup>30)</sup>にて公開されている。ダム流入量を、ダムを流域下端とした場合の流出量と同様とみなし、ダム流入量を用いて降雨流出計算の検証を行う。2017年九州北部豪雨時における寺内ダムに降雨流出モデルを適用し、その結果とダム流入量のデータを比較して、モデルの適用性を検討する。

#### 3.1.1. 解析モデル

解析モデルは、「2.3 降雨流出モデル」を適用している。

#### 3.1.2. 計算に用いたデータと条件

計算に用いたデータと条件は次の通りである。

- ・地形データ：国土地理院数値標高モデル（10 m）<sup>32)</sup>を用いている。地形モデル作成時には、JGD2011IIへの投影変換を行っており、その際にBilinear補間を行っている。図11にモデルの適用範囲と河道網を示す。集水面積は51 km<sup>2</sup>である。河道の上流端は、集水面積が0.9 ha以上となるグリッドセルとしている。
- ・河道断面：河道の断面形状は矩形に近似し、その幅 $W$ および深さ $D$ を、 $W = C_w A^{S_w}$ 、 $D = C_d A^{S_d}$ で表現する<sup>33)</sup>。ここに、 $A$ は集水面積（km<sup>2</sup>）、 $C_w$ 、 $S_w$ 、 $C_d$ 、および $S_d$ は経験的に定めるパラメーターである。
- ・降雨データ：国土交通省より提供されたXRAINデータの10分値を用いている。ただし、DIASにより提供されているXRAIN<sup>34)</sup>とは異なる。図12に、流域平均雨量の時間変化を示す。
- ・流出量データ：国土交通省水文水質データベース<sup>30)</sup>の寺内ダム流入量をダム流域からの流出量とみなして計算結果との比較に用いている。
- ・初期条件：表土層の平均水分含有率には $p_{wc}$ を与え、浸透流水深には0を与えている。

表1に計算に与えた物性値を示す。これらの値は、計算値が計測結果に近くなるように試行錯誤で定めている。

#### 3.1.3. 計算結果

図13に流出量の計算結果と観測結果の比較を示す。計算による流出量の急増する時刻や、ピーク流量、減水時間変化は、観測による値に近いことが示されている。

表 1 計算に与えた物性値

降雨流出モデル：2次元	値	流出モデル：1次元	値
空間刻み $\Delta x, \Delta y$ (m)	30	距離刻み $\Delta x$ (m)	30
表土層厚 $D$ (m)	1.0	川幅係数 $C_w$	4.73
飽和体積含水率 $\lambda^{35)}$	0.4	川幅係数 $S_w$	0.58
残留体積含水率 $p_{wc}^{35)}$	0.1	深さ係数 $C_d$	1.5
表層浸透能 $f_{s1}^{35)}$ (mm/h)	200.0	深さ係数 $S_d$	0.4
下層浸透能 $f_{s2}$ (mm/h)	0.0	マンニングの粗度係数 $n$	0.05
飽和透水係数 $k^{35)}$ (cm/s)	1.0		
等価粗度係数 $N^{35)}$	0.5		

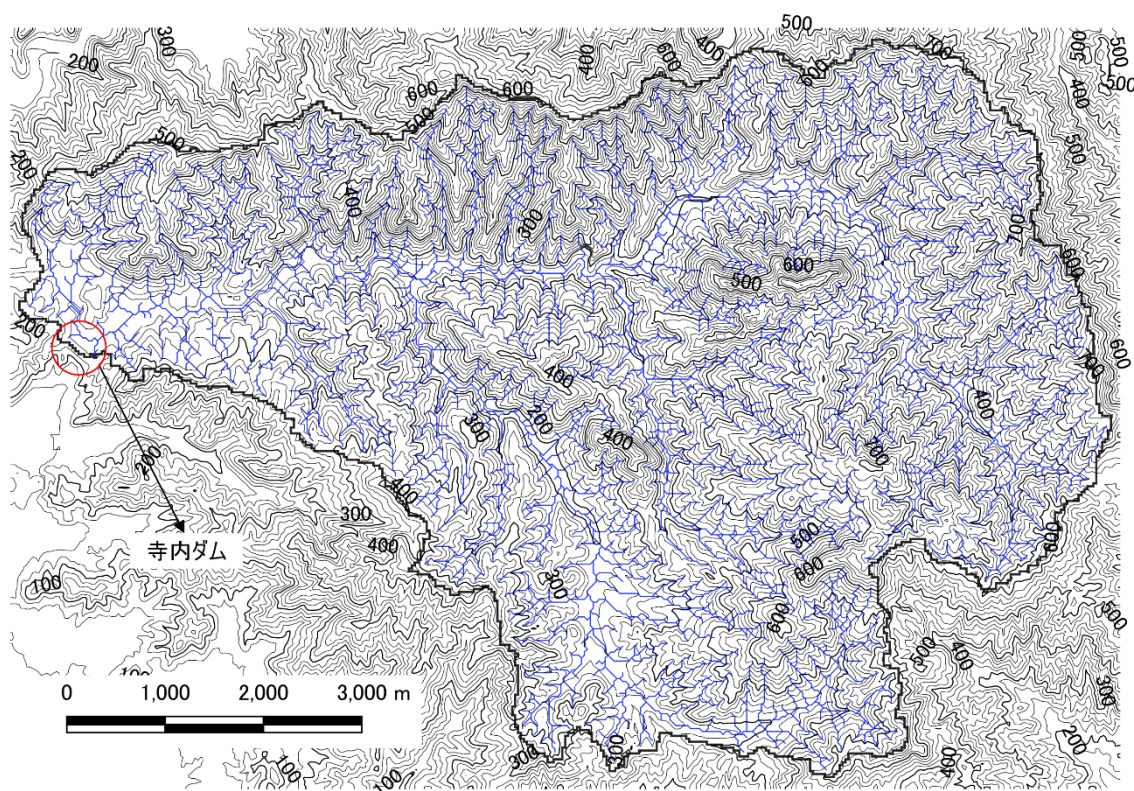


図 11 モデルの適用範囲と河道網 (国土地理院 数値標高モデルから等高線生成)

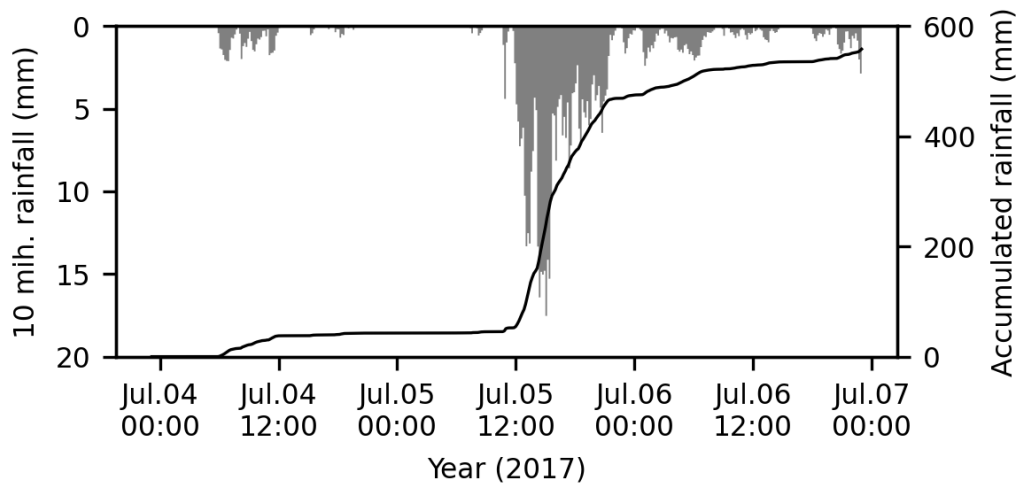


図 12 流域平均雨量の時間変化

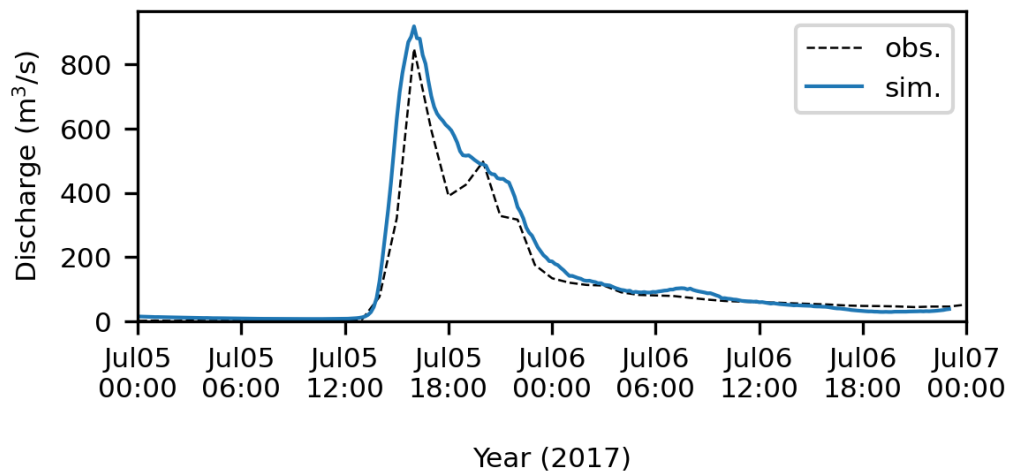


図 13 ダム流域下端の計算流出量 (sim.) と観測ダム流入量 (obs.)

### 3.2. 土石流流出

桜島有村川において2014年8月に発生した土石流に適用する。桜島有村川では、有村川3号堰堤において、土石流の表面流速、表面形状、荷重が計測されている。このデータを用いて、土石流の水深平均流速および流動深の時間変化を算出し、計算結果を比較して、モデルの適用性を検討する。

#### 3.2.1. 桜島有村川における土石流観測

##### 3.2.1.1. 対象河道

図14に有村川3号堰堤（以下、堰堤という）上流の地形図を、図15にR01、R02河道の縦断図を示す。有村川流域は桜島南東斜面に位置しており、堰堤上流の流域面積は約1.6 km<sup>2</sup>である。R01河道では、1,500 mから上流は22~33°の急勾配となっているが、R02河道では、勾配が最も急な区間でも12°である。図14には2013年10月17、26日に計測された航空レーザー測量成果と2014年10月24、26、27日に計測された標高差分値も示している。標高差分値の-0.5 mから0.5 mを除外し、標高が減少した領域を侵食域とみなすと、R01およびR02流域における2013年10月から2014年10月の土砂流出量は、それぞれ118千m<sup>3</sup>、5千m<sup>3</sup>となる。よって、堰堤における土砂は、主にR01河道により供給されているものと推察され、このR01河道を数値解析の対象範囲としている。堰堤から上流50 m区間における河床勾配は4°、堆積勾配は3~6°程度である。

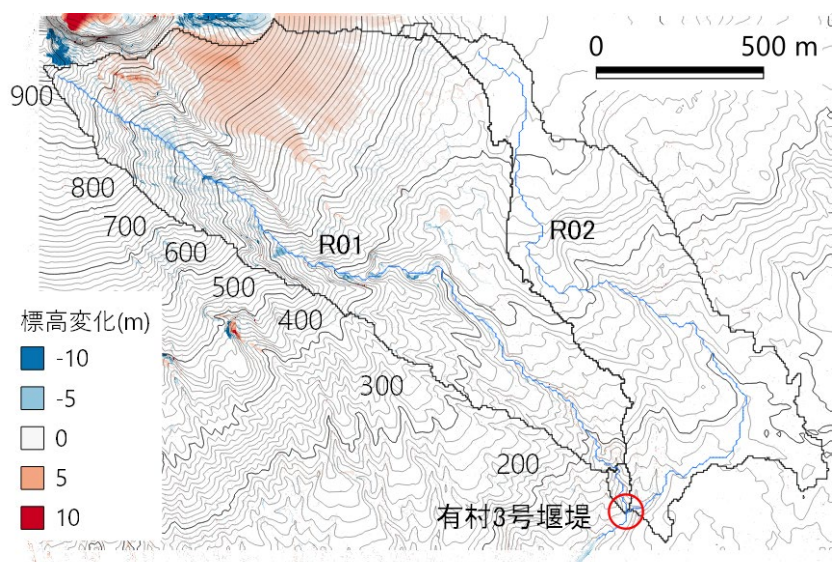


図14 有村川の位置および平面図（国土地理院 数値標高モデルから等高線生成）

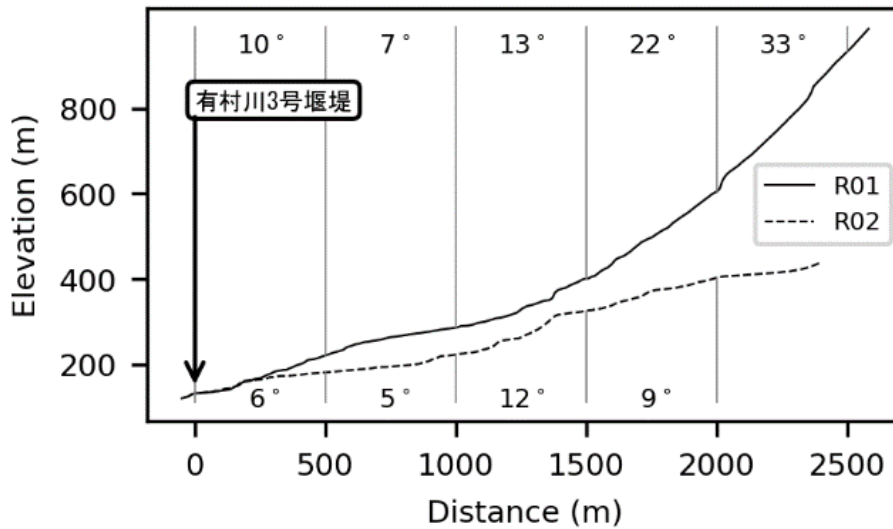


図 15 R01, R02河道の縦断面図

### 3.2.1.2. 観測項目・方法

図 16 に 2019 年 10 月 18 日に撮影した堰堤、荷重計、測域センサー、および流速計の配置状況を示す。荷重計は堰堤水通し部の左岸側に設置された幅 4 m 奥行き 2 m、厚さ 32 mm の鉄板の下部に 4 基設置されている<sup>36)</sup>。堰堤の上部には測域センサーと超音波式流速計が設置されており、測域センサーは、堰堤水通し部の横断方向の土石流の表面形状を計測している<sup>36)</sup>。荷重計、測域センサー、および超音波流速計のサンプリングレートは、それぞれ 100 Hz、20 Hz、および 1 MHz である。荷重計の分解能は 1.62 kPa/m<sup>2</sup>、測域センサーの分解能は、1 mm、0.25° /回転、流速計の分解能は 1 m/s である。これらのセンサーは独立したシステムで稼働しており、時刻も独立であるため、1~2 か月ごとのメンテナンス時に時刻補正を行なっている。





図 16 2019年10月18日に撮影した堰堤における観測機器の配置

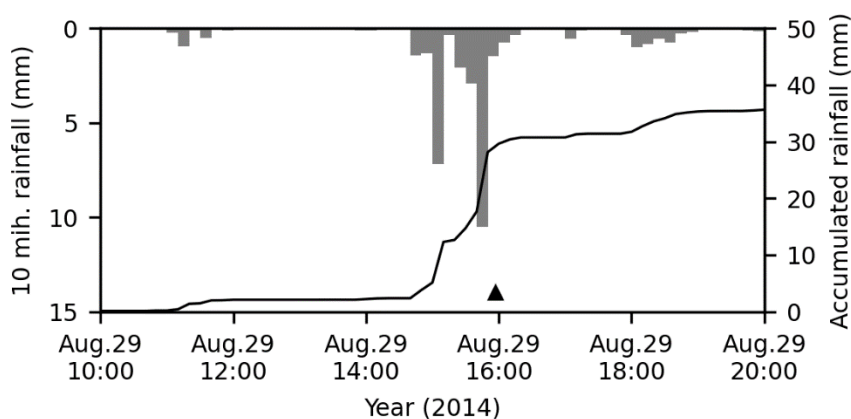


図 17 XRAINによる流域平均雨量の時間変化 (▲は土石流発生時刻を示す)

### 3.2.1.1. 観測結果

本稿では、解析の簡潔さのため、単一のピークを持ち、波形が明瞭な 2014 年 8 月 29 日 15 時 57 分 (以下 8 月 29 日) に発生した土石流を観測結果から選定し、解析の対象とする。8 月 29 日の流域平均雨量、流動深、および表面流速の時間変化は、次のとおりである。

図 17 に、XRAIN<sup>34)</sup> から作成した流域平均雨量の時間変化を示す。また、▲は土石流が発生した時刻を示している。土石流発生時刻における 10 分雨量は 10 mm で、8 月 29 日の 24 時間雨量は 36 mm である。

図 18 に 2014 年 8 月 29 日 15 時 57 分に発生した土石流の流動深 $h$ 、表面流速 $u$ 、および土石流密度 $\rho_m$ の発生時刻からの変化を示す。流動深 $h$ は、荷重計の横断幅 4 m における平均値である。土石流密度 $\rho_m$ は、計測された荷重を垂直応力値とみなして、荷重計上の土石流を

直方体近似した体積で除して算出している<sup>36)</sup>。土石流発生から7分後に流動深 $h$ 、流速 $u$ および土石流密度 $\rho_m$ がともに急増していることから、この時土石流が堰堤に到達したと考えられる。流動深 $h$ 、流速 $u$ 、および土石流密度 $\rho_m$ について、明瞭な単一のピークが見られ、ピーク出現のタイミングは、それぞれ10、11、17分であり、その値は、0.95 m、4.8 m/s、 $1.67 \text{ cm}^3/\text{s}$ である。土石流密度 $\rho_m$ の時間変化は、流動深 $h$ および流速 $u$ の時間当たりの変化よりも小さく、ピーク出現のタイミングも遅くなっている。

また、粗粒砂濃度 $c_c$ および間隙流体密度 $\rho$ を次のように推定している。堰堤における土石流の流れを平衡状態と仮定し、前述の平衡勾配の式(36)と土石流の質量密度の式(37)において平衡勾配を河床勾配に置き換えれば、土石流の粗粒砂濃度 $c_c$ および間隙流体密度 $\rho$ は、次式により表される。

$$\tan \theta_e = \frac{(\sigma/\rho - 1)c_c}{(\sigma/\rho - 1)c_c + 1} \tan \phi \quad (128)$$

$$\rho_m = (\sigma - \rho)c_c + \rho \quad (129)$$

$$c_c = \frac{\rho_m}{\sigma - (1 - \tan \theta / \tan \phi)\rho_m} \frac{\tan \theta}{\tan \phi} \quad (130)$$

$$\rho = \left(1 - \frac{\tan \theta}{\tan \phi}\right) \rho_m \quad (131)$$

図19に式(130)、(131)により算出した粗粒砂濃度 $c_c$ 、間隙流体密度 $\rho$ の発生時刻からの変化を示す。土石流密度 $\rho_m$ には観測により得られた時系列の値、河床勾配 $\theta$ には観測地の河床勾配 $4^\circ$ を入力して算出している。粗粒砂濃度 $c_c$ は、17分にピークが見られ、その値は0.14である。間隙流体密度 $\rho$ も同時刻にピークとなり、 $1.51 \text{ g/cm}^3$ である。

有村川で観測されている土石流の主な材料は、火山灰や河床の堆積物である。これらの粒度分布は、土石流の流動性に大きな影響を及ぼす、微細砂含有率や代表粒径を決定づける。図20に、火山灰と河床堆積物の粒度分布を示す。火山灰の粒度分布は、2014年10月21日および11月13日に堰堤付近で採取された試料をふるい分析および沈降分析により算出している。河床堆積物の粒度分布は、2014年10月23日に堰堤付近で採取された37.5 mm未満の試料のふるい分析および沈降分析結果、2019年10月18日に撮影した画像(図21)の分析結果と合成して算出した。

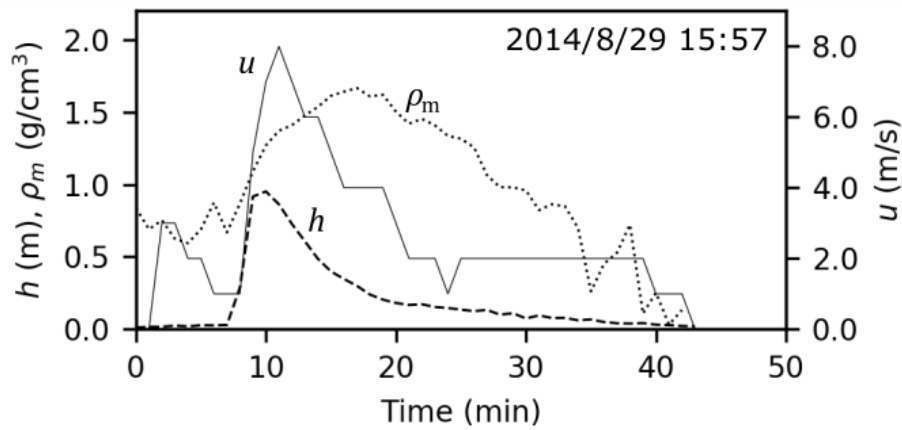


図 18 土石流発生時刻からの土石流の流動深 $h$ , 流速 $u$ , および密度 $\rho_m$ の時間変化

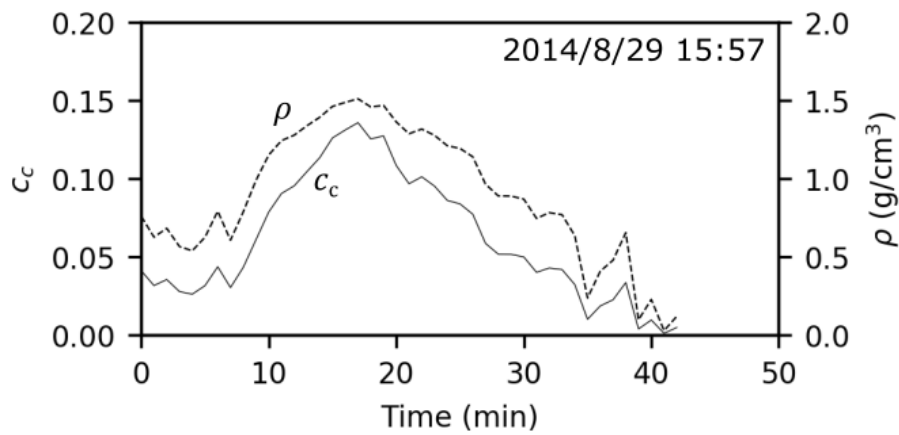


図 19 土石流発生時刻からの土石流の粗粒砂濃度 $c_c$ , および間隙流体密度 $\rho$ 時間変化

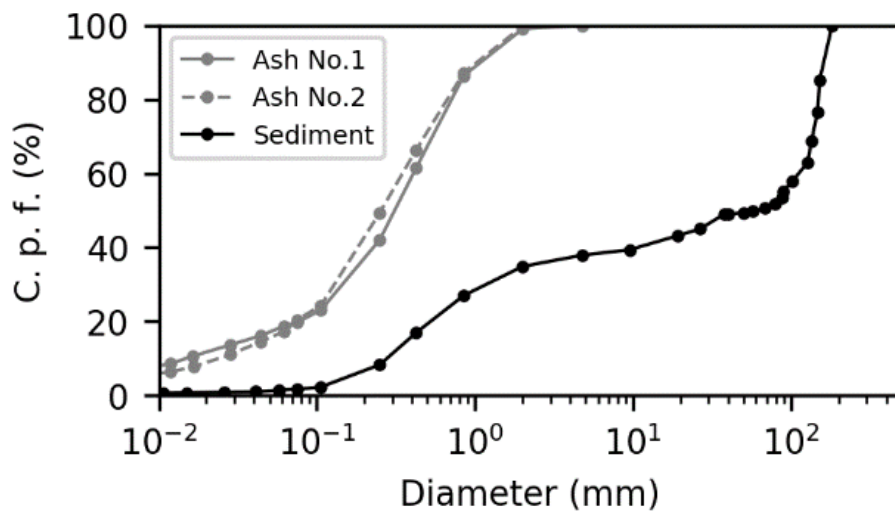


図 20 火山灰と河床堆積物の粒度分布



図 21 2019年10月18日に撮影した堰堤付近の堆積物における礫

### 3.2.2. 解析モデル

解析モデルには、「2.4 土石流流出モデル」を適用している。

### 3.2.3. 計算に用いたデータと条件

地形データについては、国土地理院数値標高モデル (10 m)<sup>32)</sup> を用いている。表 2 に計算に与えた物性値を示す。堆積物厚、代表粒径、微細砂含有率、および粘着力は、計算による流動深と流速の時間変化が観測値に近くなるように試行錯誤で定めている。斜面から河道への給水条件の実態は不明であるため、試行錯誤によって便宜的に、堆積物を水で飽和させ、河道の上流端から定常で 0.01 m<sup>3</sup>/s を与えている。

表 2 計算に与えた物性値

土石流流出モデル：1次元	値
距離刻み $\Delta x$ (m)	10
河道幅 $B$ (m)	10
堆積深 (侵食可能深) $D$ (m)	1.0
代表粒径 $d$ (m)	0.2
微細砂含有率 $p_f$	0.3
水の質量密度 $\rho_w$ <sup>19)</sup> (kg/m <sup>3</sup> )	1000
砂礫の質量密度 $\sigma$ <sup>19)</sup> (kg/m <sup>3</sup> )	2650
砂礫の内部摩擦角 $\phi$ <sup>19)</sup> (°)	35
堆積土砂濃度 $c_*$ <sup>19)</sup>	0.6
マンニングの粗度係数 $n$	0.05

### 3.2.4. 計算結果

再現計算結果より、図 22 に堰堤地点における流動深 $h$ と流速 $u$ の計算値を示す。また、図 18 に示した観測値も示している。発生源での土石流の発生時刻は不明であるため、計算値、観測値ともに、堰堤地点における流動深 $h$ 、流速 $u$ が 0 から急増してピーク値を示すことに着目し、図 18 の 8 分の値が図 22 の計算開始後 4 分の値となるように重ね合わせて、計算結果の再現性を評価している。流速 $u$ の計算値は、計算開始から 4 分後に急増し 5 分後に 4.2 m/s を示し、その後は観測値と同様に減少している。流動深 $h$ の計算値は、4 分後に急増し、5 分後に 1.4 m を示し、その後は観測値と同様に減少している。

図 23 に、土石流観測に基づく粗粒砂濃度 $c_c$ の計算値と、解析モデルによる計算値を、図 22 と同様に重ね合わせて表示している。粗粒砂濃度 $c_c$ の計算値は、4 分より 5 分にかけて、0 から 0.2 まで直線的に急増し、その後は緩やかに減少している。一方、観測値については上に凸の変化を示し、ピーク値は 0.14 を示している。

このように、再現計算では、流動深、流速ともに観測値に近い値を示し、増加・減少傾向も整合する結果が得られている。この結果は、土石流流出モデルが、実際の土石流の解析にも適用できる可能性を示している。

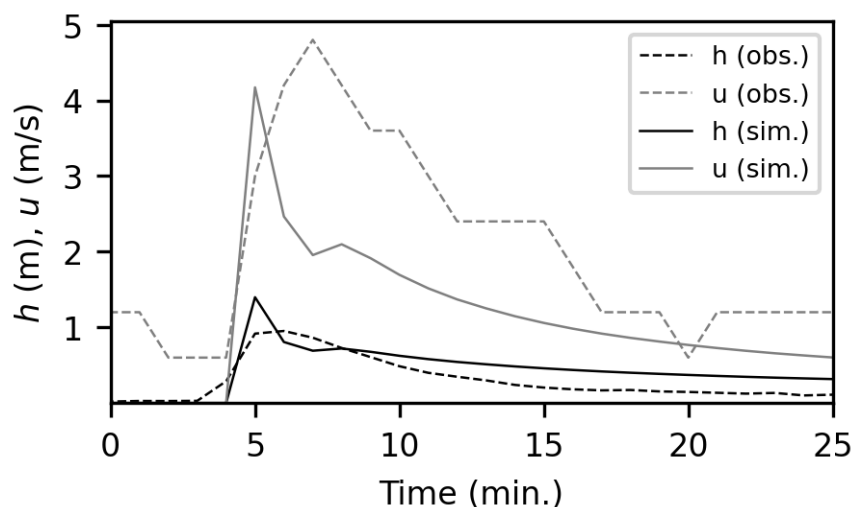


図 22 流動深 $h$ と水深平均流速 $v$ の時間変化 (obs. : 観測値、sim. : 計算値)

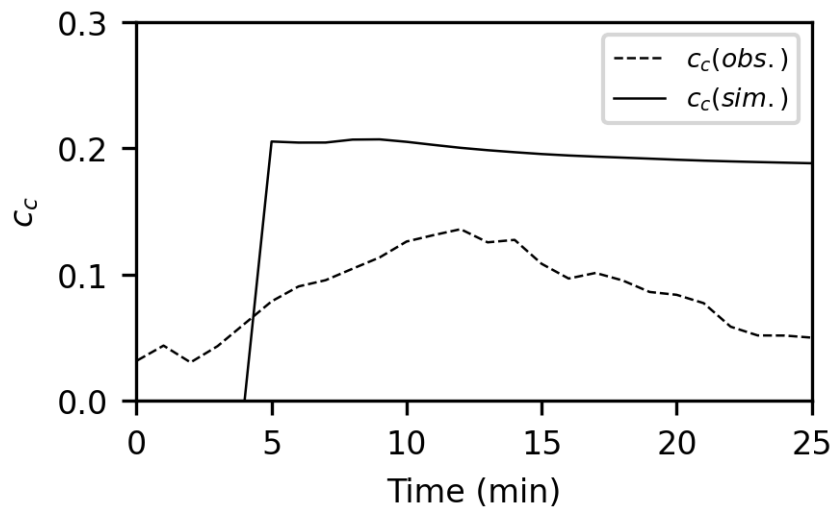


図 23 粗粒砂濃度 $c_c$ の時間変化 (obs. : 観測値、sim. : 計算値)

### 3.3. 土石流氾濫

1999年九州北部豪雨時における阿蘇古恵川の土石流に適用する。1990年7月の豪雨により、阿蘇市一の宮町では古恵川から土石流が氾濫し、土砂災害が発生した。この災害においては、災害の発生時刻、土石流の痕跡、氾濫範囲や氾濫した土砂量に関する調査結果<sup>31)</sup>がある。これらの情報を用いて、土石流流出および土石流氾濫の再現性を検討する。

#### 3.3.1. 阿蘇古恵川の土石流災害の概要

図24にアメダス阿蘇乙姫地点の1990年7月2日の時間雨量を示す。最大時間雨量は、7月2日9時から10時の67mm、24時間雨量は448mmである。坂梨地区、松原橋ともに9時20分に土石流が到達し、松原橋の土石流の流下継続時間は1時間である<sup>31)</sup>。阿蘇古恵川により氾濫した土砂の総量は36万 $\text{m}^3$ と推定されている<sup>31)</sup>。谷出口地点（後述図25○）における痕跡調査<sup>31)</sup>によると、ピーク流量は960 $\text{m}^3/\text{s}$ である。

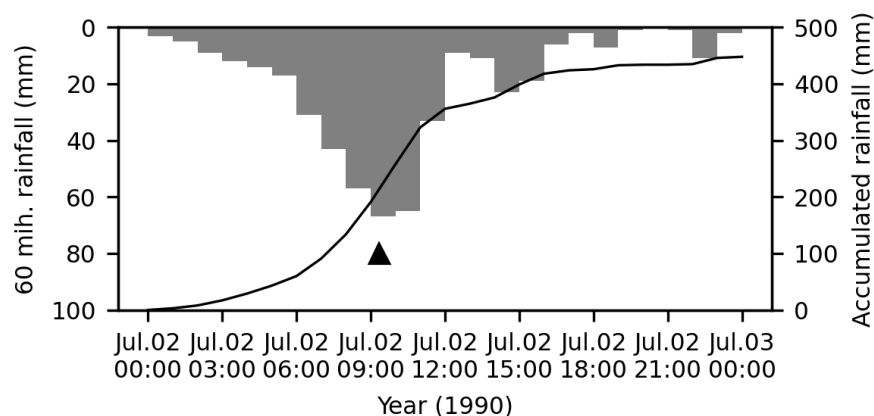


図24 アメダス阿蘇乙姫地点の時間雨量

#### 3.3.2. 解析モデル

解析モデルは、「2.3 降雨流出モデル」および「2.4 土石流流出モデル」を、氾濫が発生した範囲の上流の集水域に適用し、「2.5 土石流氾濫モデル」を氾濫範囲に適用している。図25に降雨流出モデルおよび土石流流出モデルの適用範囲、土石流氾濫モデルの適用範囲を示す。

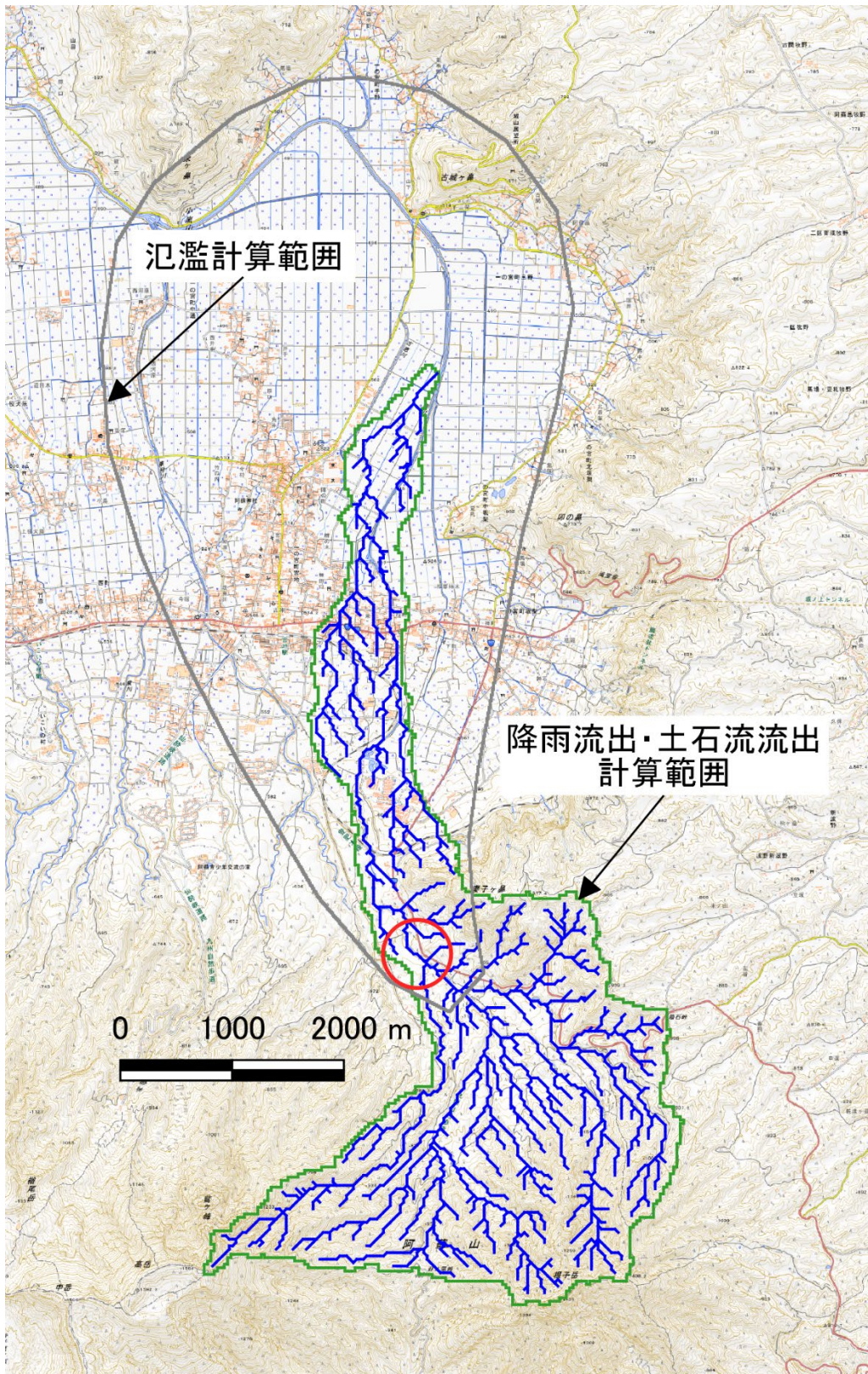


図 25 モデルの適用範囲と谷出口地点(○) (背景図に地理院地図を使用)



### 3.3.3. 計算に用いたデータと条件

計算に用いたデータと条件は次の通りである。

- ・地形データ：国土地理院数値標高モデル（10 m）<sup>32)</sup> を用いている。地形モデル作成時には、JGD2011II への投影変換を行っており、その際に Bilinear 補間を行っている。図 25 にモデルの適用範囲と河道網を示す。谷出口上流域の集水面積は 8.4 km<sup>2</sup> である。河道の上流端は、集水面積が 0.9 ha 以上となるグリッドセルとしている。
- ・河道断面：河道の断面形状は矩形に近似し、その幅  $W = 15$  m、深さ  $D = 2$  m としている。
- ・降雨データ：アメダス阿蘇乙姫地点の 1 時間データ（図 24）を計算領域全体に均等に与えている。土石流の発生・非発生を判定する雨量の時間分解能は、10 分程度が必要である<sup>37)、38)</sup> ため、土石流の現象を解析する上では、10 分データを用いることが望ましいが、本検討では災害当時のデータが存在している 1 時間データを用いている。

図 25 の○で示した点より上流では、溢流の発生を制御している。本解析モデルでは、溢流の発生する地点は、土石流の表面標高と流路側岸の地盤標高によって定まる。しかしながら、便宜的に溢流の発生を制御したい場合に、この地点を定めると、それより上流では溢流が発生しなくなり、溢流による河道流量の減少を避けることができる。再現性の高い地形モデルを作成できた場合にはこの処理は不要である。

表 3 に計算に与えた物性値を示す。参考にする資料がない場合は、現地調査<sup>31)</sup> により推定されたピーク流量 960 m<sup>3</sup>/s および流出土砂量 36 万 m<sup>3</sup> に計算値が近くなるように試行錯誤で定めている。

浸透流水深の初期条件は、次のようにして与えている。まず浸透流水深の初期値を 0 として、さらに堆積物の移動を考慮しない状態（侵食・堆積速度を常に 0 に設定した状態）で計算し、計算終了時の浸透流水深を与えている。

表 3 計算に与えた物性値

降雨流出モデル：2次元	値
空間刻み $\Delta x, \Delta y$ (m)	30
表土層厚 $D$ (m)	2.0
飽和体積含水率 $\lambda^{35)}$	0.43
残留体積含水率 $p_{wc}^{35)}$	0.045
表層浸透能 $f_{s1}^{35)}$ (mm/h)	36.0
下層浸透能 $f_{s2}$ (mm/h)	36.0
飽和透水係数 $k^{35)}$ (cm/s)	0.001
等価粗度係数 $N^{35)}$	0.1
土石流出モデル：1次元	値
距離刻み $\Delta x$ (m)	30
河道幅 $B$ (m)	15
堆積深（侵食可能深） $D$ (m)	2.0
代表粒径 $d$ (m)	0.1
微細砂含有率 $p_f$	0.4
水の質量密度 $\rho_w^{19)}$ (kg/m <sup>3</sup> )	1000
砂礫の質量密度 $\sigma^{19)}$ (kg/m <sup>3</sup> )	2650
砂礫の内部摩擦角 $\phi^{19)}$ (°)	35.0
粘着力 $c$ (N/m <sup>2</sup> )	3000
堆積土砂濃度 $c_*^{19)}$	0.6
土石流氾濫モデル：2次元	値
空間刻み $\Delta x, \Delta y$ (m)	10
堆積深 $D$ (m)	0.0
代表粒径 $d$ (m)	0.01
微細砂含有率 $p_f$	0.2
水の質量密度 $\rho_w^{19)}$ (kg/m <sup>3</sup> )	1000
砂礫の質量密度 $\sigma^{19)}$ (kg/m <sup>3</sup> )	2650
砂礫の内部摩擦角 $\phi^{19)}$ (°)	35.0
堆積土砂濃度 $c_*^{19)}$	0.6

### 3.3.4. 計算結果

図 26 に土石流流出モデルにより設定時間毎に出力された土石流流量から抽出された最大値の空間分布 (a) と、計算終了時の河床標高変化量の空間分布(b)を示す。最大流量は、設定した溢流開始点より上流 1000 m 程度の地点における  $1400 \text{ m}^3/\text{s}$  である。溢流開始点における最大流量は  $960 \text{ m}^3/\text{s}$  である。

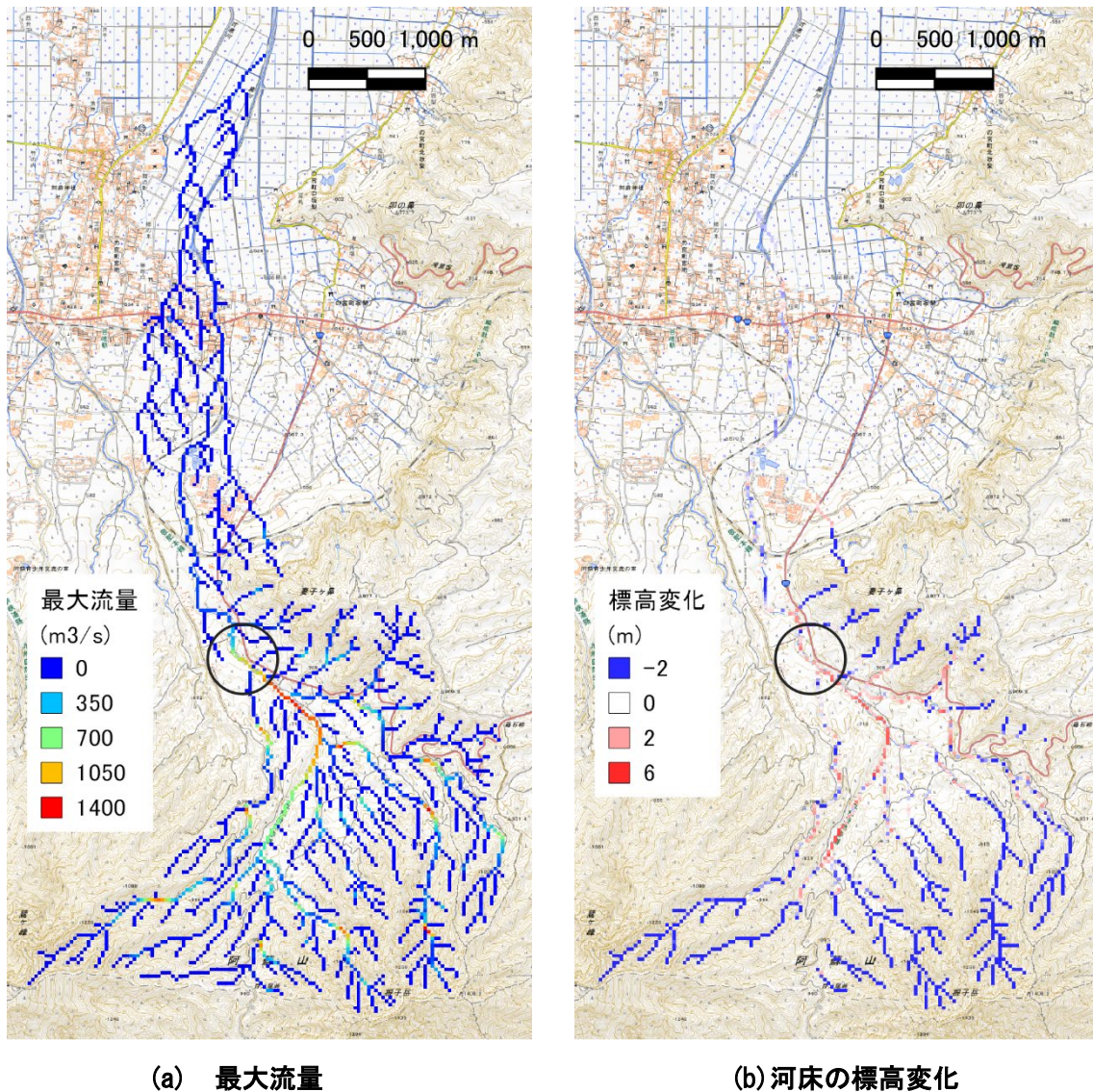


図 26 最大流量と河床の標高変化 (○は谷出口を示す) (背景図に地理院地図を使用)

#### 3.3.4.1. 土石流ハイドログラフ

図 27 に、谷出口における土石流 (水、粗粒砂、微細砂)、粗粒砂、および微細砂の流出量と累積流出量の時間変化を示す。粗粒砂と微細砂の累積流出量は、それぞれ  $78.6 \times 10^3 \text{ m}^3$ 、 $186.7 \times 10^3 \text{ m}^3$  である。これらは空隙を含まない量であるため、空隙を 0.4 とすると、合計で

442×10<sup>3</sup>m<sup>3</sup> となっている。また、土石流のピーク流量は 1050 m<sup>3</sup>/s となっており、その発生時刻は、実際に土石流が発生した時刻<sup>31)</sup>である 9 時 20 分に近い。また、ピーク流量が図 26 のものと異なるのは、図 26 の場合は出力設定時間毎の瞬間値から抽出した値であり、一方、図 27 の場合は累積値から算出した値であることから生じる違いである。

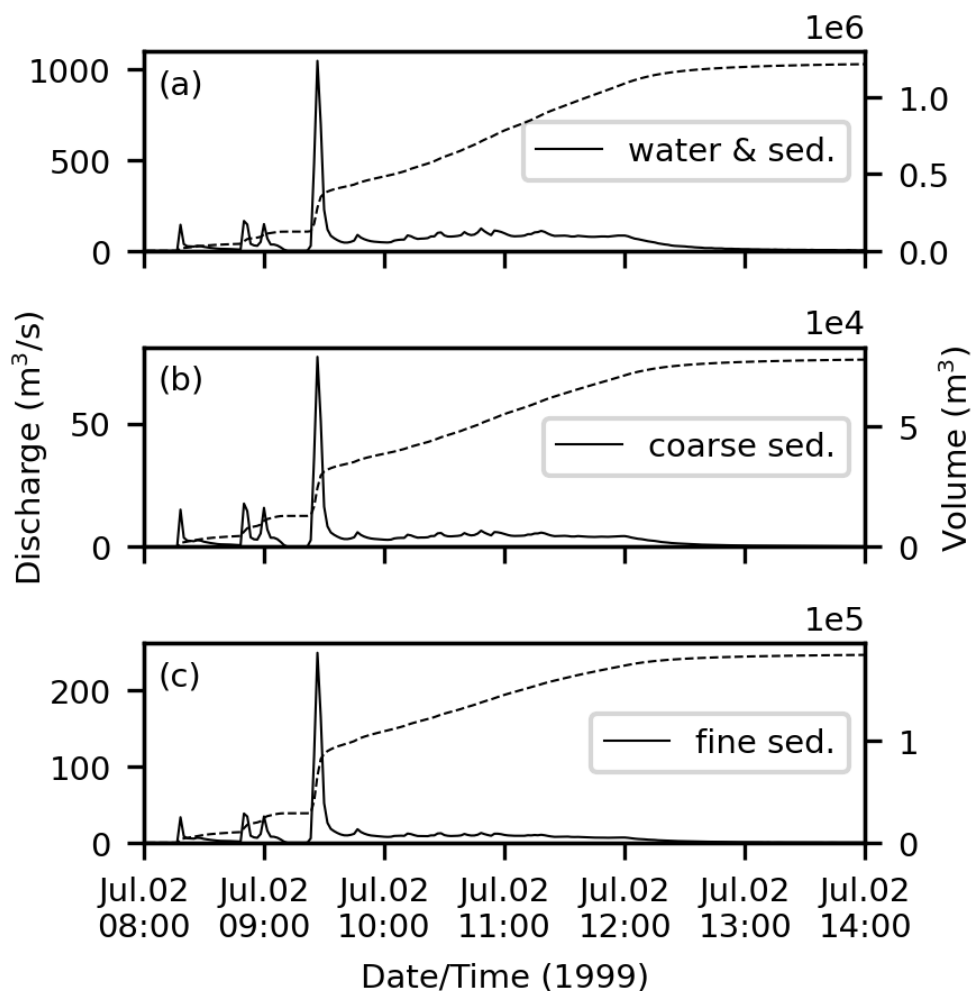


図 27 土石流のヒドログラフ（実線：時間変化、破線：累積値、  
a：水+粗粒砂+微細砂、b：粗粒砂、c：微細砂）

#### 3.3.4.2. 土石流氾濫

図 28 に最大流動深の計算値と調査結果による氾濫範囲を示す。最大流動深は 0.01 m 未満を非表示にしている。計算による流動深は、領域の北端に氾濫水が滞留している。これは、地形モデルに水路や道路盛土などを表現することで改善されるものと推定される。

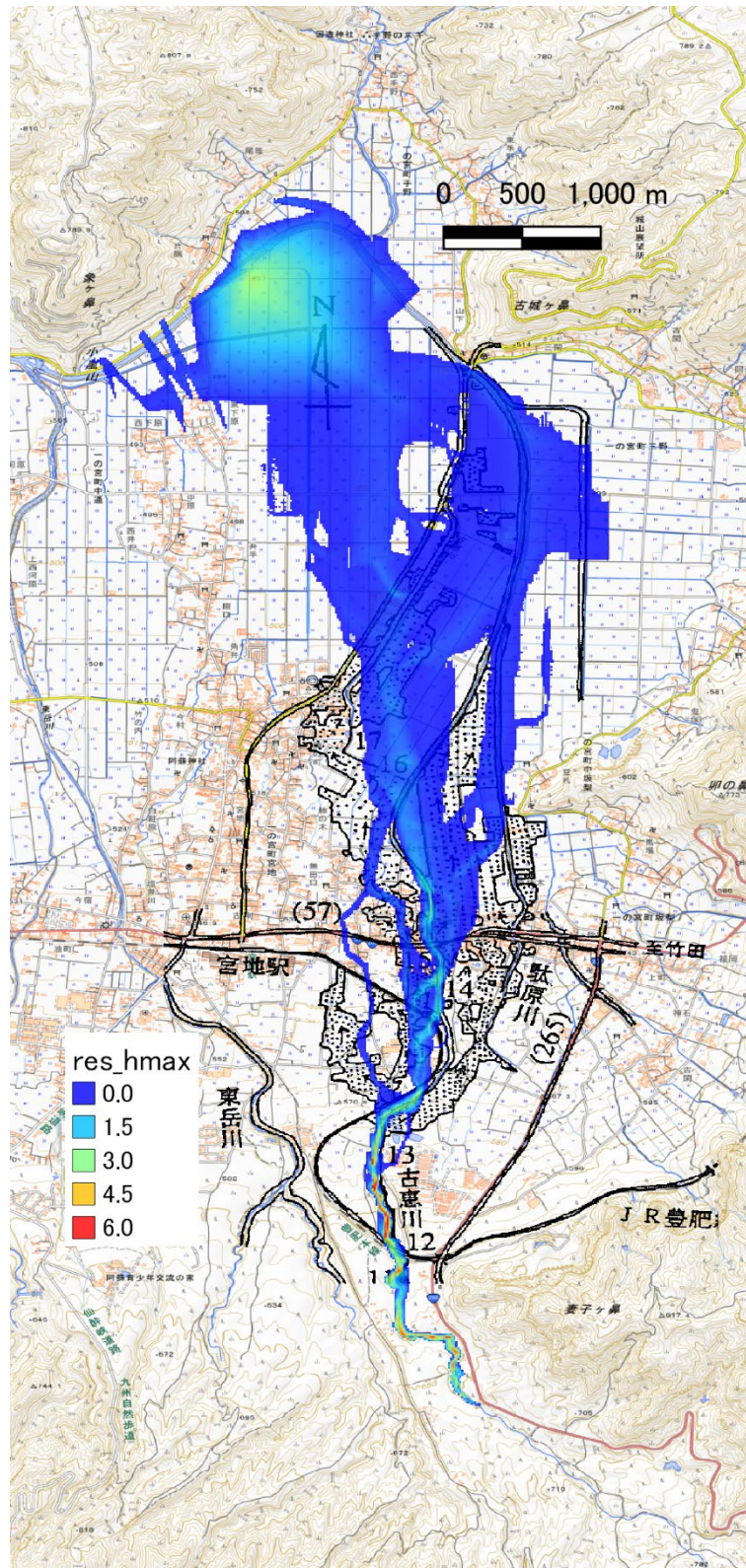


図 28 最大流動深(0.01m未満非表示)と調査結果<sup>31)</sup> (背景図に地理院地図を使用)

#### 4. おわりに

本稿では、降雨や降灰厚分布などの時間・空間分解能が高く、かつ、リアルタイム性の高い、データを活用でき、土石流の氾濫危険度評価のさらなる精度向上を図るための降雨流出-土石流流出-土石流氾濫モデルを紹介し、実際に発生した事例に適用して検証を行った。その結果、以下の点が示された。

降雨流出では、洪水流量の時間変化、土石流流出では、流速・流動深の時間変化、および土石流氾濫では、谷出口からの土石流流量および氾濫範囲についての再現性を検討した。その結果、洪水流量、土石流の流速・流動深の時間変化、および谷出口の土石流流量についての再現性は良好であった。土石流の氾濫範囲については調査結果と一致しない部分も見られた。これは、氾濫原地形において、水路は道路盛土の表現によって改善されるものと推察される。

本解析モデルにおいて、各サブモデルに適用している理論には既往の成果<sup>10),11)</sup>を活用しているが、現地で発生した水・土砂移動現象への適用事例は多くない。よって、今後、様々な事例に適用し、モデル設定の妥当性などについての検討を行う必要がある。また、このことにより、土砂災害が発生していない地域への適用における設定は、慎重に行う必要がある。

本稿で使用し、付録に添付しているプログラムソースコードは既往の成果<sup>17) 19)</sup>を編集したものである。

## 参考文献

- 1) 内田太郎, 山越隆雄, 清水武志, 吉野弘祐, 木佐洋志, 石塚忠範: 河道閉塞 (天然ダム) 及び火山の噴火を原因とする土石流による被害範囲を速やかに推定する手法, 土木技術資料, Vol.53, No.7, pp.18-23, 2011.
- 2) 清水武志, 内田太郎, 山越隆雄, 石塚忠範: 天然ダムによる土石流想定範囲計算システム (QUAD-L) の開発と 2011 年台風 12 号災害における適用, 土木技術資料, Vol.54, No.10, pp.14-17, 2012.
- 3) 高橋保・匡尚富: 変勾配流路における土石流の形成, 京都大学防災研究所年報, 29 号, B-2, pp.343-359, 1986.
- 4) 高橋保・中川一: 豪雨時に発生する石礫型土石流の予測, 砂防学会誌, 44 巻, 3 号, pp.12-19, 1991.
- 5) 中川一・高橋保・里深好文・川池健司: 1999 年ベネズエラのカムリグランデ流域で発生した土砂災害について—数値シミュレーションによる再現計算と砂防施設配置効果の評価—, 京都大学防災研究所年報, 44 号, B-2, pp.207-228, 2001.
- 6) 中谷加奈, 里深好文, 水山高久: GUI を実装した土石流一次元シミュレータ開発, 砂防学会誌, 61 巻, 2 号, pp.41-46, 2008.
- 7) 里深好文, 水山高久: 砂防ダムが設置された領域における土石流の流動・堆積に関する数値計算, 砂防学会誌, 58 巻, 1 号, pp.14-19, 2005.
- 8) 中谷加奈, 和田孝志, 里深好文, 水山高久: GUI を実装した汎用土石流シミュレータ開発, 第 4 回土砂災害に関するシンポジウム論文集, pp.149-154, 2008.
- 9) 和田孝志, 里深好文, 水山高久: 土石流計算における 1 次元・2 次元シミュレーションモデルの結合, 砂防学会誌, 61 巻, 2 号, pp.36-40, 2008.
- 10) 江頭進治, 宮本邦明, 伊藤隆郭: 掃流砂量に関する力学的解釈, 水工学論文集, Vol.41, pp.789-794, 1997.
- 11) 江頭進治: 土石流の停止・堆積のメカニズム (1), 砂防学会誌, 46 巻, 1 号, pp.45-49, 1993.
- 12) 竹林洋史, 江頭進治, 藤田正治: 2013 年 10 月に伊豆大島で発生した泥流の平面二次元解析, 河川技術論文集, 第 20 巻, pp.391-396, 2014.
- 13) 江頭進治, 伊藤隆郭: 土石流の数値シミュレーション, 日本流体力学会数値流体力学部門 Web 会誌, 第 12 巻, 第 2 号, pp.33-43, 2004.
- 14) 山崎祐介, 林真一郎, 石井靖雄: 火山噴火後に発生する土石流の氾濫リスク評価の精度向上と、降灰厚分布情報を迅速にリスク評価に反映するシステムの開発, 土木技術資料, Vol.61, No.12, pp.8-11, 2019.
- 15) 里深好文, 吉野弘祐, 小川紀一朗, 森 俊勇, 水山高久, 高濱淳一郎: 高礫山天然ダム決壊時に発生した洪水の再現, 砂防学会誌, 59 巻, 6 号, pp.32-37, 2007.
- 16) 西口幸希, 内田太郎, 田中健貴, 蒲原潤一, 奥山遼佑, 日名純也, 松原智生, 桜井 亘:

- 深層崩壊の発生に伴う土砂移動現象と被害発生位置の実態, 砂防学会誌, 68 巻, 6 号, pp.31-41, 2016.
- 17) 山崎祐介, 江頭進治, 岩見洋一: 避難予警報のための土砂災害シミュレーターに関する研究, 土木学会論文集 B1 (水工学), Vol.72, No.4, pp.I\_1327-I\_1332, 2016.
  - 18) 江頭進治, 萬矢敦啓, エスカローナ・ロシレット, 山崎祐介, 工藤 俊: 土石流形成における微細砂の役割, 平成 28 年度砂防学会研究発表会概要集, B, pp.72-73, 2016.
  - 19) 山崎祐介, 江頭進治: 豪雨に伴う土砂・流木の生産と流下過程に関する研究, 河川技術論文集, 第 24 巻, pp.71-76, 2018.
  - 20) 江頭進治, 宮本邦明, 竹林洋史: 崩壊に伴う土石流・泥流の形成と規模の決定機構, 砂防学会誌, 68 巻, 5 号, pp.38-42, 2016.
  - 21) Chen C.Y., Fujita M.: A method for predicting landslides on a basin scale using water content indicator, 土木学会論文集 B1 (水工学), Vol.70, No.4, pp.I\_13-I\_18, 2014.
  - 22) 竹林洋史, 江頭進治, 藤田正治: 2013 年 10 月に伊豆大島で発生した泥流の現地調査と数値解析, 京都大学防災研究所年報, 57 号, B, pp.372-378, 2014.
  - 23) 山崎祐介, 江頭進治, 南雲直子: 豪雨時における土砂流出量の推定法, 土木学会論文集 B1 (水工学), Vol.74, No.4, pp.I\_931-I\_936, 2018.
  - 24) 芦田和男, 江頭進治, 劉炳義: 二層モデルによる複断面河道の流れおよび河床変動の数値解析, 京都大学防災研究所年報, 第 35 号, B-2, pp.41-62, 1992.
  - 25) 国土交通省水管理・国土保全局: 河川砂防技術基準 調査編, 2014.
  - 26) 森林水文学編集委員会編: 森林水文学 森林の水のゆくえを科学する, 森北出版株式会社, 2007.
  - 27) Julien P. Y., Paris A.: Mean velocity of mudflows and debris flows, Journal of Hydraulic Engineering, Vol.136, No.9, pp.676-679, 2010.
  - 28) 日野幹雄: 明解水理学, 丸善, 1983.
  - 29) 高橋保, 中川一, 西崎丈能: 堤防決壊による洪水危険度の評価に関する研究, 京都大学防災研究所年報, 29 号, B-2, pp.431-450, 1986.
  - 30) 国土交通省水管理・国土保全局: 水文水質データベース, <http://www1.river.go.jp/>
  - 31) 橋本晴行, 平野宗夫, 林 重徳, 梅村 順: 1990 年 7 月熊本県一の宮町の土石流災害について, 水文・水資源学会誌, Vol.4, No.1, pp.25-32, 1991.
  - 32) 国土地理院基盤地図情報: <https://fgd.gsi.go.jp/download/menu.php>
  - 33) 田中茂信, 佐山敬洋: 降雨の極値統計と流出・氾濫現象の現地調査及びモデリング, 2017 年九州北部豪雨災害調査報告書, 京都大学防災研究所, pp.72-79, 2018.
  - 34) DIAS XRAIN リアルタイム雨量情報システム: <https://diasjp.net/service/xrain/>
  - 35) 田方 智, 山越隆雄, 栗原淳一, 笹原克夫, 桜庭雅明, 高橋 秀, 小野寺勝: 新規細粒火山灰が堆積した流域における分布型流出解析モデルの検討, 砂防学会誌, 60 巻, 4 号, pp.15-24, 2007.



- 36) 大坂 剛, 高橋英一, 國友 優, 山越隆雄, 能和幸範, 木佐洋志, 石塚忠範, 宇都宮玲, 横山康二, 水山高久: 桜島における土石流荷重計による単位体積重量測定, 砂防学会誌, 65 卷, 6 号, pp.46-50, 2013.
- 37) 奥田節夫: 土石流の計測法に関する研究, 京都大学防災研究所年報, 15 号, A, pp.35-41, 1972.
- 38) 池田暁彦, 水山高久, 原口勝則: 土石流の発生を支配する降雨量に関する考察, 砂防学会誌, 60 卷, 3 号, pp.26-31, 2007.

## A. 付録

ここでは、解析事例で用いた解析モデルのプログラム、入出力ファイル処理および可視化プログラム Debris Flow Simulator for Sabo (DFSS)を紹介する。プログラムの実行環境は、OSGeo4W (<https://trac.osgeo.org/osgeo4w/>) において構築される GRASS GIS、Python、GDAL が実行できる環境と FORTRAN のコンパイラである。

OSGeo4W の環境構築については、セットアップで「アドバンスインストール」を選択すると、以下のパッケージとそのバージョンが選択できる。

- GRASS GIS(7.8.6-1)
- Python(3.9.5-2)
- The GDAL/OGR Python3 Bindings and Scripts(3.3.1-1)
- Python plotting package(3.3.2-1)

FORTRAN コンパイラ

- gfortran : MinGW-w64 - for 32 and 64 bit Windows (<https://sourceforge.net/projects/mingw-w64/files/>) より、x86\_64-8.1.0-release-posix-seh-rt\_v6-rev0.7z のパッケージを利用しているが、「Fortran 90」の言語仕様であれば問題ないと思われる。

## A. 1. 寺内ダムの計算セット「Terauchi\_Dam」

フォルダ構成と各フォルダにおける作業内容は以下のとおりである。

- └01\_SET\_BASIN：計算領域の設定と標高データ・流下方向データの抽出
  - └01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル） 共通
  - └data\_from\_GSI：国土地理院の数値標高モデル(zip)の tif への変換
    - └grass：GRASS GIS 用フォルダ
- └02\_MK\_TOPO：地形モデルファイルの作成
  - └01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル） 共通
- └03\_MK\_RAIN：降雨データファイルの作成
- └04\_EXE\_SIMU：計算プログラムの実行
  - └01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル） 共通
- └05\_MK\_FIG：結果の可視化

### A. 1. 1「01\_SET\_BASIN」フォルダ

#### 1. 作業内容

国土地理院基盤地図情報ダウンロードサービスから得られる数値標高モデルから計算対象領域のデータを抽出する。

#### 2. 処理プログラムとフォルダ構成

└data\_from\_GSI:国土地理院の数値標高モデル(zip)を tif へ変換する。解析に用いた DEM ファイルは以下の通り。著作権により同梱していない。

fgddem.py(© 2012, Minoru Akagi) : xml を geotif へ変換。実行するためには、「fgddem.py.droptarget.bat」のアイコンに zip ファイルをドロップする。

FG-GML-5030-05-DEM10B.zip

FG-GML-5030-06-DEM10B.zip

FG-GML-5030-15-DEM10B.zip

FG-GML-5030-16-DEM10B.zip

└grass：GRASS GIS 用フォルダ（GRASS によって作成、以降 GRASS のみが利用）

JGD2011（GSI のデータに合わせている）

JGD2011\_II（計算領域に適した座標系を作成する）

└01\_RR\_DR

01\_latlon.py(GRASS GIS のテキストモードで実行):1 回目は「python3\_01\_latlon.py」

（コマンドは「python」でも可能な場合がある）で実行すると、DEM ファイルの読み込み、結合を行い、集水面積の分布図（acc\_ll.tif）を出力する。これをもとに対象とする流域末端のセル中心座標を取得（緯度経度）する。2 回目に「python3\_01\_latlon.py\_ [経度]\_[緯度]」（緯度\_経度の順ではないので注意）として実行すると、十進法による ddd.dxxxxxx のフォーマットで与えた緯度経度（以下、[経度]、

[緯度]のフォーマットは同様)を末端とする流域が設定され出力される (basin\_ll.tif)。これにより設定された流域を確認し、適切に設定されていない場合は、流域末端のセル中心座標の取得からやり直す。

02\_xy.py (GRASS GIS のテキストモードで実行) : 1 回目は「python3\_02\_xy.py」で実行すると、平面直角座標系への投影を行い、acc\_ll.tif を出力する。これをもとに流域末端座標を取得する。この時、データは平面直角座標系に変換されているが、流域末端座標は経度、緯度とする。2 回目に「python3\_02\_xy.py\_[経度]\_[緯度]」として実行すると、与えた緯度経度を末端とする流域が設定され、標高および流下方向ファイルが出力される。空間解像度は、02\_xy.py の内部で定義されている変数 res で指定する(サンプルは res=30 m)。

dem\_30m.asc (出力) : 標高ファイル (設定した解像度で作成される)

dir\_30m.asc (出力) : 流下 8 方向ファイル (設定した解像度で作成される)

#### A. 1. 2 「02\_MK\_TOPO」 フォルダ

##### 1) 作業内容

流域地形モデルファイルおよび河道地形モデルファイルを作成する。「01\_SET\_BASIN」で出力された標高ファイルおよび流下方向ファイルを「input」フォルダにコピーする。

##### 2) 処理プログラムとフォルダ構成

###### └01\_RR\_DR

01\_mk\_topographyFiles.f90 : 地形ファイルを作成。01.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

02\_mk\_watershedConfigurationFiles.f90 : 流域地形モデルファイルを作成。02.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

03\_mk\_streamConfigurationFiles.f90 : 河道地形モデルファイルを作成。03.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

topographyConfiguration.txt : 設定ファイル。x@outlet、y@outlet には下流端の x,y 座標を平面直角座標系で入力する。自然な地形であれば「01\_SET\_BASIN」で設定した流域末端位置の平面直角座標系の x,y の値でよいが、ダム流域の場合はダムの堤体や水面が地形としてデータ化されている場合が多いため、下流端処理の都合により、計算下流端はダム湖内部とする。

###### └input

dem\_30m.asc : 標高ファイル

dir\_30m.asc : 流下 8 方向ファイル

###### └output

targetArea\_inRR.asc : 計算対象領域ファイル

elevation\_inRR.asc : 標高ファイル  
flowDir\_inRR.asc : 流下 8 方向ファイル  
flowAcc\_inRR.asc : 集水セル数ファイル  
controlVolume\_centerPoint\_inRR.txt : スカラー計算点配列番号ファイル  
fluxPoint\_x\_inRR.txt : x 方向ベクター計算点配列番号ファイル  
fluxPoint\_xBoundary\_inRR.txt : x 方向境界点配列番号ファイル  
fluxPoint\_y\_inRR.txt : y 方向ベクター計算点配列番号ファイル  
fluxPoint\_yBoundary\_inRR.txt : y 方向境界点配列番号ファイル  
streamConfiguration\_inRR.txt : 河道構成ファイル  
volcanicAsh\_inRR.asc : 火山灰厚さファイル。便宜的に内容がすべて 0 のファイル  
をここで作成しているだけである。実際に計算する必要がある場合は別途用意  
する。

#### └misc

dir\_deg.txt : 流下 8 方向確認ファイル  
streamConfiguration\_Checker.txt : 河道網確認ファイル  
streamOrder.asc : 河道次数ファイル (Horton-Strahler)

### A. 1. 3 「03\_MK\_RAIN」フォルダ

#### 1) 作業内容

国交省により DIAS 経由で提供される XRAIN データから、解析プログラム用の降雨  
ファイルを作成する。

#### 2) 処理プログラムとフォルダ構成

##### └XRAIN\_from\_DIAS

201707040000-201707080000-10-FUK-130.7031-33.5000-130.8500-33.3688.zip : zip 圧縮  
された csv ファイル群。著作権により同梱していない。DIAS でオーダーした条件  
は次のとおり。

開始データ日時 : 2017/07/04 00:00、終了データ日時 : 2017/07/08 00:00

データ時間間隔 : 10 分

領域北西端 : 33.5000N、130.7031E、領域南東端 : 33.3688N、130.8500E

1 ファイルあたりのデータ数 : 南北 64、東西 48

ファイル形式 : CSV、圧縮形式 : zip

01\_mk\_rain.py : 雨量ファイル作成プログラム。この Python プログラムの実行には、  
Python の実行環境の設定が必要である。01.bat を実行すると、環境構築およびプログ  
ラムの実行が行われる。

targetArea\_inRR.asc (入力) : 「02\_MK\_TOPO」にて作成した計算対象領域ファイル  
rain\_time\_201707040000\_201707080000.txt (出力) : 計算領域範囲の降雨時系列ファイ

ル

#### A. 1. 4 「04\_EXE\_SIMU」 フォルダ

##### 1) 作業内容

降雨流出計算を行う。

##### 2) 処理プログラムとフォルダ構成

###### └─01\_RR\_DR

RR.bat : gfortran が実行できる環境において、コンパイル・ビルド、OpenMP のスレッド数の設定、プログラムの実行を行う。スレッド数は変更可能。

RR\_ver\_1.0.f90 : 降雨流出計算プログラム

RR\_input.txt : モデル設定ファイル

└─input : 計算入力ファイル群

→ 02\_MK\_TOPO¥01\_RR\_DR¥output のファイルをコピーする。

targetArea\_inRR.asc : 計算対象領域ファイル

elevation\_inRR.asc : 標高ファイル

flowDir\_inRR.asc : 流下 8 方向ファイル

flowAcc\_inRR.asc : 集水セル数ファイル

controlVolume\_centerPoint\_inRR.txt : スカラー計算点配列番号ファイル

fluxPoint\_x\_inRR.txt : x 方向ベクター計算点配列番号ファイル

fluxPoint\_xBoundary\_inRR.txt : x 方向境界点配列番号ファイル

fluxPoint\_y\_inRR.txt : y 方向ベクター計算点配列番号ファイル

fluxPoint\_yBoundary\_inRR.txt : y 方向境界点配列番号ファイル

streamConfiguration\_inRR.txt : 河道構成ファイル

volcanicAsh\_inRR.asc : 火山灰厚さファイル

← 02\_MK\_TOPO¥01\_RR\_DR¥output のファイルをコピーする。

rain\_Terauchi.txt : 計算領域範囲の降雨時系列ファイル : 国交省提供のデータを用いているため、「03\_MK\_RAIN」で作成したものとは異なる。

└─output\_RR : 結果ファイル群

res\_1d\_?????????.txt : 指定時間毎の 1 次元河道のファイルファイル

res\_2d\_\*\_?????????.asc : 指定時間・変数毎の 2 次元平面のファイルファイル

res\_2d\_hmax.asc : 表面流の計算期間最大値

res\_2d\_hsmax.asc : 浸透流の計算期間最大値

└─output\_misc : モデル設定確認用ファイル群

flowVolume\_RRtoDR.txt (出力) : RR モデルから DR モデルへの受渡し流量

#### A. 1. 5 「05\_MK\_FIG」 フォルダ

##### 1) 作業内容

計算結果の可視化を行う。

##### 2) 処理プログラムとフォルダ構成

RR\_out\_hyeto.py : RR モデルによる流域平均雨量のハイエトグラフを出力する。

RR\_out\_hydro.py : RR モデルによるハイドログラフを出力する。

この Python プログラムの実行には、Python の実行環境の設定が必要である。01.bat を実行すると、環境構築およびプログラムの実行が行われる。最終行のプログラム実行のコマンドの後ろにスペース区切りで出力したい計算点番号を与える。

## A. 2. 桜島有村川の計算セット「Arimura\_Riv」

フォルダ構成と各フォルダにおける作業内容は以下のとおりである。

└01\_SET\_BASIN：計算領域の設定と標高データ・流下方向データの抽出

└└01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル）共通

└└data\_from\_GSI：国土地理院の数値標高モデル(zip)の tif への変換

└└└grass：GRASS GIS 用フォルダ

└02\_MK\_TOPO：地形モデルファイルの作成

└└01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル）共通

└03\_MK\_RAIN：降雨データファイルの作成

└04\_EXE\_SIMU：計算プログラムの実行

└└01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル）共通

└05\_MK\_FIG：結果の可視化

### A. 2. 1「01\_SET\_BASIN」フォルダ

#### 1) 作業内容

国土地理院基盤地図情報ダウンロードサービスから得られる数値標高モデルから計算対象領域のデータを抽出する。

#### 2) 処理プログラムとフォルダ構成

└data\_from\_GSI:国土地理院の数値標高モデル(zip)を tif へ変換する。解析に用いた DEM ファイルは以下の通り。著作権により同梱していない。

fgddem.py(© 2012, Minoru Akagi) : xml を geotif へ変換。実行するためには、「fgddem.py.droptarget.bat」のアイコンに zip ファイルをドロップする。

FG-GML-4730-24-DEM10B.zip

FG-GML-4730-25-DEM10B.zip

FG-GML-4730-34-DEM10B.zip

FG-GML-4730-35-DEM10B.zip

└grass：GRASS GIS 用フォルダ（GRASS によって作成、以降 GRASS のみが利用）

JGD2011（GSI のデータに合わせている）

JGD2011\_II（計算領域に適した座標系を作成する）

└└01\_RR\_DR

01\_latlon.py(GRASS GIS のテキストモードで実行):1 回目は「python3\_01\_latlon.py」

（コマンドは「python」でも可能な場合がある）で実行すると、DEM ファイルの読み込み、結合を行い、集水面積の分布図（acc\_ll.tif）を出力する。これをもとに対象とする流域末端のセル中心座標を取得（緯度経度）する。2 回目に「python3\_01\_latlon.py\_ [経度]\_ [緯度]」（緯度\_経度の順ではないので注意）として実行すると、与えた緯度経度を末端とする流域が設定され出力される（basin\_ll.tif）。これ



により設定された流域を確認し、適切に設定されていない場合は、流域末端のセル中心座標の取得からやり直す。

02\_xy.py (GRASS GIS のテキストモードで実行) : 1 回目は「python3\_02\_xy.py」で実行すると、平面直角座標系への投影を行い、acc\_ll.tif を出力する。これをもとに流域末端座標を取得する。この時、データは平面直角座標系に変換されているが、流域末端座標は経度、緯度とする。2 回目に「python3\_02\_xy.py\_[経度]\_[緯度]」として実行すると、与えた緯度経度を末端とする流域が設定され、標高および流下方向ファイルが出力される。空間解像度は、02\_xy.py の内部で定義されている変数 res で指定する(サンプルは res=10 m)。

dem\_10m.asc (出力) : 標高ファイル (設定した解像度で作成される)

dir\_10m.asc (出力) : 流下 8 方向ファイル (設定した解像度で作成される)

## A. 2. 2 「02\_MK\_TOPO」フォルダ

### 1) 作業内容

流域地形モデルファイルおよび河道地形モデルファイルを作成する。「01\_SET\_BASIN」で出力された標高ファイルおよび流下方向ファイルを「input」フォルダにコピーする。

### 2) 処理プログラムとフォルダ構成

#### └01\_RR\_DR

01\_mk\_topographyFiles.f90 : 地形ファイルを作成。01.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

02\_mk\_watershedConfigurationFiles.f90 : 流域地形モデルファイルを作成。02.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

03\_mk\_03\_mk\_streamConfigurationFiles\_Arimura.f90 : 河道地形モデルファイルを作成。03.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。有村川は本川のみ計算を行う為、河道上流端の検出に特別処理を行っている。

topographyConfiguration.txt : 設定ファイル。x@outlet、y@outlet には下流端の x,y 座標を平面直角座標系で入力する。

#### └input

dem\_10m.asc : 標高ファイル

dir\_10m.asc : 流下 8 方向ファイル

#### └output

targetArea\_inRR.asc : 計算対象領域ファイル

elevation\_inRR.asc : 標高ファイル

flowDir\_inRR.asc : 流下 8 方向ファイル

flowAcc\_inRR.asc : 集水セル数ファイル

controlVolume\_centerPoint\_inRR.txt : スカラー計算点配列番号ファイル  
fluxPoint\_x\_inRR.txt : x 方向ベクター計算点配列番号ファイル  
fluxPoint\_xBoundary\_inRR.txt : x 方向境界点配列番号ファイル  
fluxPoint\_y\_inRR.txt : y 方向ベクター計算点配列番号ファイル  
fluxPoint\_yBoundary\_inRR.txt : y 方向境界点配列番号ファイル  
streamConfiguration\_inRR.txt : 河道構成ファイル  
volcanicAsh\_inRR.asc : 火山灰厚さファイル。便宜的に内容がすべて 0 のファイル  
をここで作成しているだけである。実際に計算する必要がある場合は別途用意  
する。

#### └misc

dir\_deg.txt : 流下 8 方向確認ファイル  
streamConfiguration\_Checker.txt : 河道網確認ファイル  
streamOrder.asc : 河道次数ファイル (Horton–Strahler)

### A. 2. 3 「03\_MK\_RAIN」フォルダ

#### 1) 作業内容

降雨を与える代わりに、上流端から一定流量を流入させるファイルを作成する。

#### 2) 処理プログラムとフォルダ構成

01\_mk\_fv.py : 流量作成ファイル雨量ファイル作成プログラム。この Python プログラム  
の実行には、Python の実行環境の設定が必要である。01.bat を実行すると、環境構築  
およびプログラムの実行が行われる。

streamConfiguration\_inRR.txt (入力) : 「02\_MK\_TOPO」で作成されたファイル。

flowVolume\_RRtoDR.txt (出力) : 計算で用いる上流端からの流量データ。

#### A. 2. 4 「04\_EXE\_SIMU」 フォルダ

##### 1) 作業内容

土石流流出計算を行う。

##### 2) 処理プログラムとフォルダ構成

###### └─01\_RR\_DR

DR.bat : gfortran が実行できる環境において、コンパイル・ビルド、OpenMP のスレッド数の設定、プログラムの実行を行う。スレッド数は変更可能。

DR\_ver\_1.0.f90 : 土石流流出計算プログラム

DR\_input.txt : モデル設定ファイル

└─input : 計算入力ファイル群

→ 02\_MK\_TOPO¥01\_RR\_DR¥output のファイルをコピーする。

streamConfiguration\_inRR.txt : 河道構成ファイル

← 02\_MK\_TOPO¥01\_RR\_DR¥output のファイルをコピーする。

└─output\_DR : 結果ファイル群

res\_?????????.txt : 指定時間毎の 1 次元河道のデータファイル

flowVolume\_RRtoDR.txt (入力) : RR-DR モデルへの受渡し流量。「03\_MK\_RAIN」で作成したもの。

flowVolume\_ws\_DRtoDF.txt (出力) : DR-DF モデルへの受渡し (水・粗粒砂・微細砂)

flowVolume\_cs\_DRtoDF.txt (出力) : DR-DF モデルへの受渡し (粗粒砂)

flowVolume\_fs\_DRtoDF.txt (出力) : DR-DF モデルへの受渡し (微細砂)

#### A. 2. 5 「05\_MK\_FIG」 フォルダ

##### 1) 作業内容

計算結果の可視化を行う。

##### 2) 処理プログラムとフォルダ構成

DR\_out\_hu.py : DR モデルによる流動深と流速を出力する。

DR\_out\_cc.py : DR モデルによる粗粒土砂濃度を出力する。

この Python プログラムの実行には、Python の実行環境の設定が必要である。01.bat を実行すると、環境構築およびプログラムの実行が行われる。最終行のプログラム実行のコマンドの後ろにスペース区切りで出力したい計算点番号を与える。

### A. 3. 阿蘇古恵川の計算セット「Furuegawa\_Riv」

フォルダ構成と各フォルダにおける作業内容は以下のとおりである。

- └01\_SET\_BASIN：計算領域の設定と標高データ・流下方向データの抽出
  - └01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル）共通
  - └02\_DF：土石流氾濫モデル（DF モデル）
  - └data\_from\_GSI：国土地理院の数値標高モデル(zip)の tif への変換
  - └grass：GRASS GIS 用フォルダ
- └02\_MK\_TOPO：地形モデルファイルの作成
  - └01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル）共通
  - └02\_DF：土石流氾濫モデル（DF モデル）
- └03\_MK\_RAIN：降雨データファイルの作成
- └04\_EXE\_SIMU：計算プログラムの実行
  - └01\_RR\_DR：降雨流出モデル（RR モデル）・土石流流出モデル（DR モデル）共通
  - └02\_DF：土石流氾濫モデル（DF モデル）
- └05\_MK\_FIG：結果の可視化

#### A. 3. 1 「01\_SET\_BASIN」フォルダ

##### 1) 作業内容

国土地理院基盤地図情報ダウンロードサービスから得られる数値標高モデルから計算対象領域のデータを抽出する。

##### 2) 処理プログラムとフォルダ構成

└data\_from\_GSI:国土地理院の数値標高モデル(zip)を tif へ変換する。解析に用いた DEM ファイルは以下の通り。著作権により同梱していない。

fgddem.py(© 2012, Minoru Akagi) : xml を geotif へ変換。実行するためには、「fgddem.py.droptarget.bat」のアイコンに zip ファイルをドロップする。

FG-GML-4931-20-DEM10B.zip

FG-GML-4931-21-DEM10B.zip

FG-GML-4931-30-DEM10B.zip

FG-GML-4931-31-DEM10B.zip

└grass：GRASS GIS 用フォルダ（GRASS によって作成、以降 GRASS のみが利用）

JGD2011（GSI のデータに合わせている）

JGD2011\_II（計算領域に適した座標系を作成する）

└01\_RR\_DR

01\_latlon.py (GRASS GIS のテキストモードで実行) : 1 回目は「python3\_01\_latlon.py」

（コマンドは「python」でも可能な場合がある）で実行すると、DEM ファイルの読み込み、結合を行い、集水面積の分布図（acc\_ll.tif）を出力する。これをもとに

対象とする流域末端のセル中心座標を取得（緯度経度）する。2 回目に「python3\_01\_latlon.py」[経度][緯度]（緯度\_経度の順ではないので注意）として実行すると、与えた緯度経度を末端とする流域が設定され出力される（basin\_ll.tif）。これにより設定された流域を確認し、適切に設定されていない場合は、流域末端のセル中心座標の取得からやり直す。

02\_xy.py（GRASS GIS のテキストモードで実行）：1 回目は「python3\_02\_xy.py」で実行すると、平面直角座標系への投影を行い、acc\_ll.tif を出力する。これをもとに流域末端座標を取得する。この時、データは平面直角座標系に変換されているが、流域末端座標は経度、緯度とする。2 回目に「python3\_02\_xy.py」[経度][緯度]として実行すると、与えた緯度経度を末端とする流域が設定され、標高および流下方向ファイルが出力される。空間解像度は、02\_xy.py の内部で定義されている変数 res で指定する(サンプルは res=30 m)。

dem\_30m.asc（出力）：標高ファイル（設定した解像度で作成される）

dir\_30m.asc（出力）：流下 8 方向ファイル（設定した解像度で作成される）

#### └02\_DF

02\_xy\_01.py（GRASS GIS のテキストモードで実行）：「python3\_02\_xy\_01.py」と実行すると、area.shp の領域が設定され、標高および流下方向ファイルが出力される。空間解像度は、02\_xy\_01.py の内部で定義されている変数 res で指定する(サンプルは res=10 m)。

area.shp（入力）：氾濫範囲指定ファイル（GIS などで作成する）土石流の氾濫量は、RR モデルおよび DR モデルで計算されるのでこれらの領域を含む（すべて包括する必要はない）ように作成する。

basin\_10m.asc（出力）：計算対象領域ファイル

dem\_10m.asc（出力）：標高ファイル（設定した解像度で作成される）

dir\_10m.asc（出力）：流下 8 方向ファイル（設定した解像度で作成される）

### A. 3. 2 「02\_MK\_TOPO」フォルダ

#### 1) 作業内容

流域地形モデルファイルおよび河道地形モデルファイルを作成する。「01\_SET\_BASIN」で出力された標高ファイルおよび流下方向ファイルを「input」フォルダにコピーする。

#### 2) 処理プログラムとフォルダ構成

##### └01\_RR\_DR

01\_mk\_topographyFiles.f90：地形ファイルを作成。01.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

02\_mk\_watershedConfigurationFiles.f90：流域地形モデルファイルを作成。02.bat にて

コンパイル・ビルド・実行。gfortran が実行できる環境が必要。

03\_mk\_streamConfigurationFiles.f90 : 河道地形モデルファイルを作成。03.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

topographyConfiguration.txt : 設定ファイル。x@outlet、y@outlet には下流端の x,y 座標を平面直角座標系で入力する。

└input

dem\_30m.asc : 標高ファイル

dir\_30m.asc : 流下 8 方向ファイル

└output

targetArea\_inRR.asc : 計算対象領域ファイル

elevation\_inRR.asc : 標高ファイル

flowDir\_inRR.asc : 流下 8 方向ファイル

flowAcc\_inRR.asc : 集水セル数ファイル

controlVolume\_centerPoint\_inRR.txt : スカラー計算点配列番号ファイル

fluxPoint\_x\_inRR.txt : x 方向ベクター計算点配列番号ファイル

fluxPoint\_xBoundary\_inRR.txt : x 方向境界点配列番号ファイル

fluxPoint\_y\_inRR.txt : y 方向ベクター計算点配列番号ファイル

fluxPoint\_yBoundary\_inRR.txt : y 方向境界点配列番号ファイル

streamConfiguration\_inRR.txt : 河道構成ファイル

volcanicAsh\_inRR.asc : 火山灰厚さファイル。便宜的に内容がすべて 0 のファイルをここで作成しているだけである。実際に計算する必要がある場合は別途用意する。

└misc

dir\_deg.txt : 流下 8 方向確認ファイル

streamConfiguration\_Checker.txt : 河道網確認ファイル

streamOrder.asc : 河道次数ファイル (Horton-Strahler)

└02\_DF

01\_mk\_floodplainConfigurationFiles.f90 : 氾濫原地形モデルファイルを作成。01.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

02\_mk\_streamFloodplainConectionFiles.f90 : 河道との空間結合ファイルを作成。02.bat にてコンパイル・ビルド・実行。gfortran が実行できる環境が必要。

topographyConfiguration.txt : 設定ファイル

└input (入力)

basin.asc : 計算対象領域ファイル

dem.asc : 標高ファイル

dir.asc : 流下 8 方向ファイル

streamConfiguration\_inRR.txt : 01\_RR\_DR で作成したデータ

#### ↳output

targetArea\_inDF.asc : 計算対象領域データ

elevation\_inDF.asc : 標高データ

flowDir\_inDF.asc : 最急勾配 8 方向データ

controlVolumeCenterPoint\_inDF.txt : スカラー計算点配列番号データ

fluxPoint\_xBoudary\_inDF.txt : x 方向境界点配列番号データ

fluxPoint\_x\_inDF.txt : x 方向ベクター計算点配列番号データ

fluxPoint\_yBoudary\_inDF.txt : y 方向境界点配列番号データ

fluxPoint\_y\_inDF.txt : y 方向ベクター計算点配列番号データ

fluxPoint\_eBoudary\_inDF.txt : x 方向 (東側) 境界点配列番号データ

fluxPoint\_nBoudary\_inDF.txt : y 方向 (北側) 境界点配列番号データ

fluxPoint\_sBoudary\_inDF.txt : -y 方向 (南側) 境界点配列番号データ

fluxPoint\_wBoudary\_inDF.txt : -x 方向 (西側) 境界点配列番号データ

streamFloodplainConnection.txt : 河道との空間結合ファイルを作成

d.asc : 堆積厚データ\*

hs\_ini.asc : 浸透流水深データ\*

\*便宜的に内容がすべて 0 のファイルをここで作成しているだけである。実際に計算する必要がある場合は、別途用意する。ただし、ファイルのヘッダー情報はここで作成されるものと同様にする。

### A. 3. 3 「03\_MK\_RAIN」フォルダ

#### 1) 作業内容

気象庁アメダス阿蘇乙姫地点における 1990 年 7 月 2 日の時間データから解析プログラム用の降雨データを作成する。

#### 2) 処理プログラム

特になし。書式に従って rain\_Aso.txt を作成する。

### A. 3. 4 「04\_EXE\_SIMU」 フォルダ

#### 1) 作業内容

降雨流出計算を行う。

#### 2) 処理プログラムとフォルダ構成

##### └01\_RR\_DR

RR.bat : gfortran が実行できる環境において、コンパイル・ビルド、OpenMP のスレッド数の設定、プログラムの実行を行う。スレッド数は変更可能。

RR\_ver\_1.0.f90 : 降雨流出計算プログラム

RR\_input.txt : モデル設定ファイル

##### └input : 計算入力ファイル群

→ 02\_MK\_TOPO¥01\_RR\_DR¥output のファイルをコピーする。

targetArea\_inRR.asc : 計算対象領域ファイル

elevation\_inRR.asc : 標高ファイル

flowDir\_inRR.asc : 流下 8 方向ファイル

flowAcc\_inRR.asc : 集水セル数ファイル

controlVolume\_centerPoint\_inRR.txt : スカラー計算点配列番号ファイル

fluxPoint\_x\_inRR.txt : x 方向ベクター計算点配列番号ファイル

fluxPoint\_xBoundary\_inRR.txt : x 方向境界点配列番号ファイル

fluxPoint\_y\_inRR.txt : y 方向ベクター計算点配列番号ファイル

fluxPoint\_yBoundary\_inRR.txt : y 方向境界点配列番号ファイル

streamConfiguration\_inRR.txt : 河道構成ファイル

volcanicAsh\_inRR.asc : 火山灰厚さファイル

← 02\_MK\_TOPO¥01\_RR\_DR¥output のファイルをコピーする。

rain\_Aso.txt : 計算領域範囲の降雨時系列ファイル : 03\_MK\_RAIN で作成

##### └output\_RR : 結果ファイル群

res\_1d\_?????????.txt : 指定時間毎の 1 次元河道のファイルファイル

res\_2d\_\*\_?????????.asc : 指定時間・変数毎の 2 次元平面のファイルファイル

res\_2d\_hmax.asc : 表面流の計算期間最大値

res\_2d\_hsmax.asc : 浸透流の計算期間最大値

##### └output\_misc : モデル設定確認用ファイル群

flowVolume\_RRtoDR.txt (出力) : RR モデルから DR モデルへの受渡し流量



DR.bat : gfortran が実行できる環境において、コンパイル・ビルド、OpenMP のスレッド数の設定、プログラムの実行を行う。スレッド数は変更可能。

DR\_ver\_1.0.f90 : 土石流流出計算プログラム

DR\_input.txt : モデル設定ファイル

└input : 計算入力ファイル群

→ 02\_MK\_TOPO¥01\_RR\_DR¥output のファイルをコピーする。

streamConfiguration\_inRR.txt : 河道構成ファイル

← 02\_MK\_TOPO¥01\_RR\_DR¥output のファイルをコピーする。

└output\_DR : 結果ファイル群

res\_?????????.txt : 指定時間毎の 1 次元河道のデータファイル

flowVolume\_RRtoDR.txt (入力) : RR-DR モデルへの受渡し流量。「03\_MK\_RAIN」で作成したもの。

flowVolume\_ws\_DRtoDF.txt (出力) : DR-DF モデルへの受渡し (水・粗粒砂・微細砂)

flowVolume\_cs\_DRtoDF.txt (出力) : DR-DF モデルへの受渡し (粗粒砂)

flowVolume\_fs\_DRtoDF.txt (出力) : DR-DF モデルへの受渡し (微細砂)

└02\_DF

DF.bat : gfortran が実行できる環境において、コンパイル・ビルド、OpenMP のスレッド数の設定、プログラムの実行を行う。スレッド数は変更可能。

DF\_ver\_1.0.f90 : 土石流氾濫計算プログラム

DF\_input.txt : モデル設定ファイル

└input : 計算入力ファイル群

→ 02\_MK\_TOPO¥02\_DF¥output のファイルをコピーする。

targetArea\_inDF.asc : 計算対象領域データ

elevation\_inDF.asc : 標高データ

flowDir\_inDF.asc : 最急勾配 8 方向データ

controlVolumeCenterPoint\_inDF.txt : スカラー計算点配列番号データ

fluxPoint\_xBoundary\_inDF.txt : x 方向境界点配列番号データ

fluxPoint\_x\_inDF.txt : x 方向ベクター計算点配列番号データ

fluxPoint\_yBoundary\_inDF.txt : y 方向境界点配列番号データ

fluxPoint\_y\_inDF.txt : y 方向ベクター計算点配列番号データ

fluxPoint\_eBoundary\_inDF.txt : x 方向 (東側) 境界点配列番号データ

fluxPoint\_nBoundary\_inDF.txt : y 方向 (北側) 境界点配列番号データ

fluxPoint\_sBoundary\_inDF.txt : -y 方向 (南側) 境界点配列番号データ

fluxPoint\_wBoundary\_inDF.txt : -x 方向 (西側) 境界点配列番号データ

streamFloodplainConnection.txt : 河道との空間結合ファイルを作成

d.asc : 堆積厚データ  
 hs\_ini.asc : 浸透流水深データ  
 ← 02\_MK\_TOPO¥02\_DF¥output のファイルをコピーする。  
 → 01\_RR\_DR の DR モデルの計算結果ファイルをコピーする。  
 flowVolume\_ws\_DRtoDF.txt  
 flowVolume\_cs\_DRtoDF.txt  
 flowVolume\_fs\_DRtoDF.txt  
 ← 01\_RR\_DR の DR モデルの計算結果ファイルをコピーする。  
 streamConfiguration\_inRR.txt (RR モデル、DR モデルと共通)  
 rain\_Aso.txt (RR モデルと共通)  
 ↳output\_DF : 結果ファイル群  
 res\_\*\_?????????.asc : 指定時間・変数毎の 2 次元平面のデータファイル  
 res\_hmax.asc : 流動深の最大値

### A. 3.5 「05\_MK\_FIG」フォルダ

#### 1) 作業内容

計算結果の可視化を行う。

#### 2) 処理プログラムとフォルダ構成

RR\_out\_hyeto.py : RR モデルによる流域平均雨量のハイエトグラフを出力する。

RR\_out\_hydro.py : RR モデルによるハイドログラフを出力する。

DR\_out\_hydro.py : DR モデルによる水+土砂、粗粒土砂、微細土砂のハイドログラフを出力する。

この Python プログラムの実行には、Python の実行環境の設定が必要である。

01\_RR.bat を実行すると、「RR\_out\_hyeto.py」および「RR\_out\_hydro.py」の環境構築およびプログラムの実行が行われ、02\_DR.bat を実行すると、「DR\_out\_hydro.py」の実行が行われる。最終行のプログラム実行のコマンドの後ろにスペース区切りで出力したい計算点番号を与える。

↳mk\_map : DR モデルの計算結果を平面にマッピングする。

map.py : DR モデルの計算結果を平面にマッピングする。

qmax.asc : 計算期間における流量の最大値

hmax.asc : 計算期間における流動深の最大値

umax.asc : 計算期間における流速の最大値

dz.asc : 計算終了時の侵食・堆積深

r\_prof.py : DR モデルの計算結果の縦断図を出力する。

↳mk\_vtk : RR、DR、DF モデルの出力から vtk ファイルを作成する。

vtk\_line\_RR.py : DR のラインベクターファイルの作成プログラム

vtk\_line\_DR.py : DR のラインベクターファイルの作成プログラム

vtk\_cell\_DF.py : RR および DF のラスターファイルの作成プログラム

RR.bat、DR.bat、DF.bat をそれぞれ実行すると、「vtk\_line\_RR.py」、

「vtk\_line\_DR.py」、「vtk\_cell\_DF.py」の環境構築およびプログラムの実行が行われる。最終行のプログラム実行のコマンドの後ろにスペース区切りで出力したい時間間隔（秒）を与える。

↳out : 作成された vtk ファイルの保存先

# B. 付録 プログラムソースコード Debris Flow Simulator for Sabo (DFSS)

Public Works Research Institute (P.W.R.I.)\*

## 目次

1	01.SET_BASIN	B-1
1.1	01.RR_DR . . . . .	B-1
1.2	02.DF . . . . .	B-4
2	02.MK_TOPO	B-5
2.1	01.RR_DR . . . . .	B-5
2.2	02.DF . . . . .	B-33
3	03.MK_RAIN	B-49
4	04.EXE_SIMU	B-52
4.1	01.RR_DR . . . . .	B-52
4.2	02.DF . . . . .	B-97
5	05.MK_FIG	B-126
5.1	mk_map . . . . .	B-133
5.2	mk_vtk . . . . .	B-135

---

\* 火山・土石流チーム

# 1 01\_SET\_BASIN

## 1.1 01\_RR\_DR

01\_latlon.py

```
1 import sys
2 import os
3 import glob
4 files=glob.glob(os.path.join('..','data_from_GSI','FG-GML-????-??-DEM10B.tif'))
5 res=10
6 print('input -> ',sys.argv)
7 fdem='dem_%dm'%res
8 if len(sys.argv)==1:
9     fns=[]
10    for fn0 in files:
11        fn=os.path.basename(fn0)
12        print(fn)
13        cmd='r.in.gdal_in=%s_out=%s_o'%(fn0,fn[:-4])
14        os.system(cmd)
15        fns.append(fn[:-4])
16    cmd='g.region_rast=%s'%(','.join(fns))
17    os.system(cmd)
18    cmd='r.patch_in=%s_out=%s_o'%(','.join(fns),fdem)
19    os.system(cmd)
20    cmd='r.watershed_a_ele=%s_drain=dir_acc=acc_o'%fdem
21    os.system(cmd)
22    cmd='r.out.gdal_in=acc_out=acc_ll.tif_o'
23    os.system(cmd)
24    print('Step 1, output -> acc_ll.tif')
25 elif len(sys.argv)==3:
26     lon=sys.argv[1]
27     lat=sys.argv[2]
28     try:
29         x=float(lon)
30         y=float(lat)
31         cmd='r.water.outlet_input=dir_output=basin_coordinates=%f,%f_o'%(x,
32         y)
33         os.system(cmd)
34         cmd='r.out.gdal_in=basin_out=basin_ll.tif_o'
35         os.system(cmd)
36         cmd='r.to.vect_in=basin_out=basin_type=area_o'
37         os.system(cmd)
38         cmd='v.buffer_in=basin_out=basin_buf_dis=%f_o'%(3./3600)# map unit
39         os.system(cmd)
40         print('Step 2, set outlet -> (x,y)={},{}'.format(x,y))
41         print('Step 2, output -> basin_ll.tif')
42     except:
43         print('Format_Error@(x,y) -> exit')
44         exit()
45 else:
46     print('Error@input? -> exit')
```

```

1 import sys
2 import os
3 import glob
4 def trs(lon, lat, src_EPSG, dst_EPSG):
5     from osgeo import ogr, osr, gdal
6     src_srs, dst_srs = osr.SpatialReference(), osr.SpatialReference()
7     src_srs.ImportFromEPSG(src_EPSG)
8     dst_srs.ImportFromEPSG(dst_EPSG)
9     trans = osr.CoordinateTransformation(src_srs, dst_srs)
10    return trans.TransformPoint(lat, lon)[: -1][1:3] # lat, lon -> lon, lat
11 res_ll=10
12 res=30
13 loc='JGD2011'
14 print('input ->', sys.argv)
15 fdem='dem_%dm'%res_ll
16 if len(sys.argv)==1:
17     cmd='v.proj_in=%s_out=area_loc=%s_o'%( 'basin_buf', loc)
18     os.system(cmd)
19     cmd='g.region_a_vect=area_res=%d'%(res)
20     os.system(cmd)
21     cmd='r.proj_in=%s_out=dem_%dm_loc=%s_method=bilinear_o'%(fdem, res, loc)
22     os.system(cmd)
23     cmd='r.watershed_a_ele=dem_%dm_acc=acc_drain=dir_o'%res
24     os.system(cmd)
25     cmd='r.fill_dir_input=dem_%dm_dir=dir_output=dem_filled_%dm_o'%(res,
26     res)
27     os.system(cmd)
28     cmd='r.watershed_a_ele=dem_filled_%dm_acc=acc_drain=dir_o'%res
29     os.system(cmd)
30     cmd='r.out_gdal_in=acc_out=acc_xy.tif_o'
31     os.system(cmd)
32     print('Step_1_output -> acc_xy.tif')
33 elif len(sys.argv)==3:
34     #x=sys.argv[1]
35     #y=sys.argv[2]
36     ### ->
37     lon=sys.argv[1]
38     lat=sys.argv[2]
39     ### <-
40     try:
41         #x=float(x)
42         #y=float(y)
43         ### ->
44         lon=float(lon)
45         lat=float(lat)
46         (x,y)=trs(lon, lat, 6668, 6670)
47         print('{{},{}} -> -({},{})'.format(lon, lat, x, y))
48         ### <-
49         cmd='g.region_a_vect=area_res=%d'%(res)
50         os.system(cmd)
51         dem='dem_filled_%dm'%res
52         cmd='r.water.outlet_input=dir_output=basin_coordinates=%f,%f_o'%(x,
53         y)

```

```

52     os.system(cmd)
53     cmd='r.buffer_in=basin_out=basin_buf_dis=%d_o'%res
54     os.system(cmd)
55     cmd='g.region_zoom=basin_buf'
56     os.system(cmd)
57     for f in [[ 'dir', 'dir_%dm'%res, 0], [dem, 'dem_%dm'%res, -999]]:
58         fi=f[0]
59         fo=f[1]
60         nv=f[2]
61         os.system('r.out.gdal_in=%s_out=%s.asc_format=AALGrid_nodata=%d_o
--o'%(fi, fo, nv))
62         print('Step_2,_set_outlet_<--_(x,y)={},_{}'.format(x,y))
63         print('Step_2,_output_-->_dem_%dm.asc,_dir_%dm.asc'%(res,res))
64     except:
65         print('Format_Error@(x,y)_-->_exit')
66         exit()
67 else:
68     print('Error@input?_-->_exit')

```

---

## 1.2 02\_DF

02\_xy\_01.py

---

```
1 import os
2 res_ll=10
3 res=10
4 area='area.shp'
5 loc='JGD2011'
6 fdem='dem_%dm'%res_ll
7 cmd='v.in.ogr_in=%s_out=area_DF_o'%(area)
8 os.system(cmd)
9 cmd='v.buffer_in=area_DF_out=area_buf_dis=%d_o'%(res)
10 os.system(cmd)
11 cmd='g.region_a_vect=area_buf_res=%d'%(res)
12 os.system(cmd)
13 cmd='v.to.rast_in=area_DF_out=area_DF_use=val_val=1_o'
14 os.system(cmd)
15 fdem2='dem_%dm'%res
16 cmd='r.proj_in=%s_out=%s_loc=%s_method=bilinear_o'%(fdem,fdem2,loc)
17 os.system(cmd)
18 cmd='r.watershed_a_ele=%s_acc=acc_drain=dir_o'%(fdem2)
19 os.system(cmd)
20 for f in [['dir','dir_%dm'%res,0],[fdem2,'dem_%dm'%res,-999],['area_DF','basin_%dm'%res,0]]:
21     fi=f[0]
22     fo=f[1]
23     nv=f[2]
24     os.system('r.out.gdal_in=%s_out=%s.asc_format=AAIGrid_nodata=%d_o'%(fi,fo,nv))
25 exit()
```

---



## 2 02\_MK\_TOPO

### 2.1 01\_RR\_DR

01\_mk\_topographyFiles.f90

```
1 implicit none
2 integer, allocatable :: ibasin(:, :), iacc(:, :), idir(:, :), ibasin2(:, :), iacc2(:, :), &
3     i_2d(:, :), j_2d(:, :), id_2d(:, :)
4 integer, allocatable :: i_1d(:), j_1d(:), i_out(:), j_out(:)
5 integer :: iend, jend, i, j, k, i1, j1, i2, j2, itmp, itr, i0, j0, iout, jout
6 integer :: n_1d, n_out, n_ij_out, l
7 real(8), allocatable :: z_2d(:, :), facc(:, :)
8 real(8) :: xllcorner, yllcorner, cellsize, x_out, y_out
9 character(len=100) :: fname, ctmp, i_fmt, f_fmt
10 !
11 open(1, file='topographyConfiguration.txt')
12 read(1, *) iend
13 read(1, *) jend
14 allocate( ibasin(iend, jend), iacc(iend, jend), idir(iend, jend), ibasin2(iend, jend), iacc2(iend,
15     jend), &
16     i_2d(iend, jend), j_2d(iend, jend), id_2d(iend, jend), z_2d(iend, jend), facc(iend, jend))
17 read(1, *)
18 read(1, *) fname !dem
19 call r_fasc(10, fname, z_2d, iend, jend, xllcorner, yllcorner, cellsize)
20 fname='output/elevation_inRR.asc'
21 !write(ctmp, *) n_out
22 !write(i_fmt, *) len(trim(adjustl(ctmp)))+1
23 !i_fmt='*(i'//trim(adjustl(i_fmt))//)'
24 f_fmt='*(f10.3)'
25 !call w_iasc(20, fname, iriver(:, :), iend, jend, xllcorner, yllcorner, cellsize, 0)
26 call w_fasc_fmt(20, fname, z_2d, f_fmt, iend, jend, xllcorner, yllcorner, cellsize, 0.d0)
27 read(1, *) fname !dir
28 call r_iasc(10, fname, idir, iend, jend, xllcorner, yllcorner, cellsize)
29 idir=abs(idir)
30 !where(idir==0)idir=8
31 !where(idir==128)idir=0
32 !
33 ibasin=1
34 !
35 n_ij_out=1
36 allocate(i_out(n_ij_out), j_out(n_ij_out))
37 read(1, *) x_out
38 read(1, *) y_out
39 i_out=nint((x_out-xllcorner-0.5*cellsize)/cellsize)+1
40 j_out=nint((y_out-yllcorner-0.5*cellsize)/cellsize)+1
41 !
42 !! ==> find outlet
43 !n_ij_out=0
44 !do i=1, iend
45 ! do j=1, jend
46 ! if(1<=ibasins(i, j))then
47 ! call f_next(iend, jend, idir, i, j, i1, j1)
48 ! if(ibasins(i1, j1)==0)then
```

```

48 !       write(*,'(a,2i5)')'outlet(i,j)=' ,i ,j
49 !       iout=i
50 !       jout=j
51 !       n_ij_out=n_ij_out+1
52 !       end if
53 !     end if
54 ! end do
55 !end do
56 !allocate(i_out(n_ij_out),j_out(n_ij_out))
57 !!
58 !n_ij_out=0
59 !do i=1,iend
60 ! do j=1,jend
61 !   if(1<=ibasin(i,j))then
62 !     call f_next(iend,jend,idir,i,j,i1,j1)
63 !     if(ibasin(i1,j1)==0)then
64 !       n_ij_out=n_ij_out+1
65 !       i_out(n_ij_out)=i
66 !       j_out(n_ij_out)=j
67 !     end if
68 !   end if
69 ! end do
70 !end do
71 !! <— find outlet
72 !
73 i_2d(:,:)=0
74 j_2d(:,:)=0
75 id_2d(:,:)=0
76 ibasin2(:,:)=0
77 iacc2(:,:)=0
78 itr=1
79 do i=1,iend
80   do j=1,jend
81     i_2d(i,j)=i
82     j_2d(i,j)=j
83     id_2d(i,j)=(i-1)*jend+j
84     if(itr/=(i-1)*jend+j)then
85       write(*,*) itr ,(i-1)*jend+j
86       read(*,*)
87     end if
88     itr=itr+1
89   end do
90 end do
91 !
92 n_1d=count(1<=ibasin(:,:))
93 allocate(i_1d(n_1d),j_1d(n_1d))
94 i_1d(1:n_1d)=pack(i_2d(:,:),(1<=ibasin(:,:)))
95 j_1d(1:n_1d)=pack(j_2d(:,:),(1<=ibasin(:,:)))
96 !write(*,'(a,i18)')' no. of data=',n_1d
97 !
98 do k=1,n_1d
99   itmp=0
100  i0=i_1d(k)
101  j0=j_1d(k)
102  i=i_1d(k)

```

```

103  j=j_1d(k)
104  itr=1
105  i2=0
106  j2=0
107  do while(itmp==0)
108    iacc2(i,j)=iacc2(i,j)+1
109    call f_next(iend,jend,idir,i,j,i1,j1)
110    do l=1,n_ij_out
111      if(i==i_out(l) .and. j==j_out(l))then
112        ibasin2(i0,j0)=1
113        itmp=1
114        !cycle
115      end if
116    end do
117    if(i1<=1 .or. iend<=i1 .or. j1<=1 .or. jend<=j1)then
118      itmp=1
119    end if
120    i2=i
121    j2=j
122    i=i1
123    j=j1
124    itr=itr+1
125  end do
126 end do ! k
127 !
128 deallocate(i_1d,j_1d)
129 !
130 fname='output/targetArea_inRR.asc'
131 write(ctmp,*)n_out
132 write(i_fmt,*)len(trim(adjustl(ctmp)))+1
133 i_fmt='*(i'//trim(adjustl(i_fmt))//')'
134 !call w_iasc(20,fname,iriver(:,:),iend,jend,xllcorner,yllcorner,cellsize,0)
135 call w_iasc_fmt(20,fname,ibasins2,i_fmt,iend,jend,xllcorner,yllcorner,cellsize,0)
136 !
137 fname='output/flowAcc_inRR.asc'
138 write(ctmp,*)maxval(iacc2)
139 write(i_fmt,*)len(trim(adjustl(ctmp)))+1
140 i_fmt='*(i'//trim(adjustl(i_fmt))//')'
141 !call w_iasc(20,fname,iriver(:,:),iend,jend,xllcorner,yllcorner,cellsize,0)
142 call w_iasc_fmt(20,fname,iacc2,i_fmt,iend,jend,xllcorner,yllcorner,cellsize,0)
143 !
144 fname='output/flowDir_inRR.asc'
145 write(ctmp,*)maxval(idir)
146 write(i_fmt,*)len(trim(adjustl(ctmp)))+1
147 i_fmt='*(i'//trim(adjustl(i_fmt))//')'
148 !call w_iasc(20,fname,iriver(:,:),iend,jend,xllcorner,yllcorner,cellsize,0)
149 call w_iasc_fmt(20,fname,idir,i_fmt,iend,jend,xllcorner,yllcorner,cellsize,0)
150 !
151 fname='output/volcanicAsh_inRR.asc'
152 write(ctmp,*)maxval(idir)
153 write(f_fmt,*)len(trim(adjustl(ctmp)))+1
154 f_fmt='*(f10.3))'
155 !call w_iasc(20,fname,iriver(:,:),iend,jend,xllcorner,yllcorner,cellsize,0)
156 call w_fasc_fmt(20,fname,dbble(idir)*0.d0,f_fmt,iend,jend,xllcorner,yllcorner,cellsize,0.d0)
157 !

```

```

158 write(*,'(a)')'---_nomal_end_---'
159 stop
160 end
161 !
162 !
163 !
164 subroutine b_up(iend,jend,iriver,idir,i,j,itmp)
165 implicit none
166 integer :: i,j,iend,jend,itmp
167 integer :: iriver(iend,jend),idir(iend,jend)
168 !
169 ! nw n ne ! 3 2 1
170 ! \ | / ! \ | /
171 ! w-+- e ! 4-+- 8
172 ! / | \ ! / | \
173 ! sw s se ! 5 6 7
174 !if(iriver(i, j+1)==1)itmp=itmp+1 ! n
175 !if(iriver(i+1,j+1)==1)itmp=itmp+1 ! ne
176 !if(iriver(i+1, j)==1)itmp=itmp+1 ! e
177 !if(iriver(i+1,j-1)==1)itmp=itmp+1 ! se
178 !if(iriver(i, j-1)==1)itmp=itmp+1 ! s
179 !if(iriver(i-1,j-1)==1)itmp=itmp+1 ! sw
180 !if(iriver(i-1, j)==1)itmp=itmp+1 ! w
181 !if(iriver(i-1,j+1)==1)itmp=itmp+1 ! nw
182 if(idir(i-1,j-1)==1 .and. iriver(i-1,j-1)==1)itmp=itmp+1 ! 5 -> 1
183 if(idir(i, j-1)==2 .and. iriver(i, j-1)==1)itmp=itmp+1 ! 6 -> 2
184 if(idir(i+1,j-1)==3 .and. iriver(i+1,j-1)==1)itmp=itmp+1 ! 7 -> 3
185 if(idir(i+1,j) ==4 .and. iriver(i+1,j) ==1)itmp=itmp+1 ! 8 -> 4
186 if(idir(i+1,j+1)==5 .and. iriver(i+1,j+1)==1)itmp=itmp+1 ! 1 -> 5
187 if(idir(i, j+1)==6 .and. iriver(i, j+1)==1)itmp=itmp+1 ! 2 -> 6
188 if(idir(i-1,j+1)==7 .and. iriver(i-1,j+1)==1)itmp=itmp+1 ! 3 -> 7
189 if(idir(i-1,j) ==8 .and. iriver(i-1,j) ==1)itmp=itmp+1 ! 4 -> 8
190 !
191 end subroutine b_up
192 !
193 !
194 !
195 subroutine f_next(iend,jend,idir,i,j,i1,j1)
196 implicit none
197 integer :: i,j,iend,jend,i1,j1,idir(iend,jend)
198 !
199 ! nw n ne ! 3 2 1
200 ! \ | / ! \ | /
201 ! w-+- e ! 4-+- 8
202 ! / | \ ! / | \
203 ! sw s se ! 5 6 7
204 if( idir(i,j)==1)then
205 i1=i+1
206 j1=j+1
207 else if(idir(i,j)==2)then
208 i1=i
209 j1=j+1
210 else if(idir(i,j)==3)then
211 i1=i-1
212 j1=j+1

```

```

213 else if(idir(i,j)==4)then
214   i1=i-1
215   j1=j
216 else if(idir(i,j)==5)then
217   i1=i-1
218   j1=j-1
219 else if(idir(i,j)==6)then
220   i1=i
221   j1=j-1
222 else if(idir(i,j)==7)then
223   i1=i+1
224   j1=j-1
225 else if(idir(i,j)==8)then
226   i1=i+1
227   j1=j
228 !else if(idir(i,j)<=-1 .or. iriver(i,j)==0)then
229 ! write(*,'(a,4i5)')'river end @ i,j,dir,iriv=',i,j,idir(i,j),iriver(i,j)
230 ! itmp=1
231 else if(idir(i,j)==0)then
232   i1=-1
233   j1=-1
234 else
235   write(*,'(a,4i5)')'river_course?_@_i,j,dir,iriv=',i,j,idir(i,j) !,iriver(i,j)
236   write(*,*)'Don't_come_here?'
237   write(*,*)'iend,jend',iend,jend
238   stop
239 end if
240 !
241 end subroutine f_next
242 !
243 !
244 !
245 subroutine r_iasc(fon,fname,ia,iend,jend,xllcorner,yllcorner,cellsize)
246 implicit none
247 integer :: j,iend,jend,fon,ncols,nrows,ia(1:iend,1:jend)
248 real(8) :: xllcorner,yllcorner,cellsize
249 character(len=100) :: ctmp,fname
250 !
251 open(fon,file=trim(adjustl(fname)))
252 read(fon,*)ctmp,ncols
253 read(fon,*)ctmp,nrows
254 if(ncols /= iend .or. nrows /= jend) then
255   write(*,*)'Check_ncols_/_iend_or_nrows_/_jend'
256   stop
257 end if
258 read(fon,*)ctmp,xllcorner
259 read(fon,*)ctmp,yllcorner
260 read(fon,*)ctmp,cellsize
261 read(fon,*)
262 do j=jend,1,-1
263   read(fon,*)ia(1:iend,j)
264 end do
265 close(fon)
266 !
267 end subroutine r_iasc

```

```

268 !
269 !
270 !
271 subroutine r_fasc(fon , fname , a , iend , jend , xllcorner , yllcorner , cellsize )
272 implicit none
273 integer :: j , iend , jend , fon , ncols , nrows
274 real(8) :: xllcorner , yllcorner , cellsize , a(1:iend , 1:jend )
275 character(len=100) :: ctmp , fname
276 !
277 open(fon , file=trim(adjustl(fname)))
278 read(fon , *) ctmp , ncols
279 read(fon , *) ctmp , nrows
280 if (ncols /= iend .or. nrows /= jend) then
281   write(* , *) 'Check_ncols_/=_iend_or_nrows_/=_jend'
282   stop
283 end if
284 read(fon , *) ctmp , xllcorner
285 read(fon , *) ctmp , yllcorner
286 read(fon , *) ctmp , cellsize
287 read(fon , *)
288 do j=jend , 1 , -1
289   read(fon , *) a(1:iend , j)
290 end do
291 close(fon)
292 !
293 end subroutine r_fasc
294 !
295 !
296 !
297 subroutine w_iasc(fon , fname , ia , iend , jend , xllcorner , yllcorner , cellsize , nv)
298 implicit none
299 integer :: j , iend , jend , fon , nv , ia(1:iend , 1:jend)
300 real(8) :: xllcorner , yllcorner , cellsize
301 character(len=100) :: fname , ciend
302 !
303 write(ciend , *) iend
304 open(fon , file=trim(adjustl(fname)))
305 write(fon , '(a,i5)') 'ncols' , iend
306 write(fon , '(a,i5)') 'nrows' , jend
307 write(fon , '(a,f15.3)') 'xllcorner' , xllcorner
308 write(fon , '(a,f15.3)') 'yllcorner' , yllcorner
309 write(fon , '(a,f10.3)') 'cellsize_' , cellsize
310 write(fon , '(a,i5)') 'NODATA_value' , nv
311 do j=jend , 1 , -1
312   write(fon , '(*(i2)') ia(1:iend , j)
313 end do
314 close(fon)
315 !
316 end subroutine w_iasc
317 !
318 !
319 !
320 subroutine w_iasc_fmt(fon , fname , ia , str , iend , jend , xllcorner , yllcorner , cellsize , nv)
321 implicit none
322 integer :: j , iend , jend , fon , nv , ia(1:iend , 1:jend)

```

```

323 real(8) :: xllcorner, yllcorner, cellsize
324 character(len=100) :: fname, ciend, str
325 !
326 write(ciend,*)iend
327 open(fon, file=trim(adjustl(fname)))
328 write(fon, '(a,i5)')'ncols', iend
329 write(fon, '(a,i5)')'nrows', jend
330 write(fon, '(a,f15.3)')'xllcorner', xllcorner
331 write(fon, '(a,f15.3)')'yllcorner', yllcorner
332 write(fon, '(a,f10.3)')'cellsize_', cellsize
333 write(fon, '(a,i5)')'NODATA_value', nv
334 do j=jend,1,-1
335     write(fon, str)ia(1:iend, j)
336 end do
337 close(fon)
338 !
339 end subroutine w_iasc_fmt
340 !
341 !
342 !
343 subroutine w_fasc(fon, fname, a, iend, jend, xllcorner, yllcorner, cellsize, nv)
344 implicit none
345 integer :: j, iend, jend, fon, nv
346 real(8) :: xllcorner, yllcorner, cellsize, a(1:iend, 1:jend)
347 character(len=100) :: fname, ciend
348 !
349 write(ciend,*)iend
350 open(fon, file=trim(adjustl(fname)))
351 write(fon, '(a,i5)')'ncols', iend
352 write(fon, '(a,i5)')'nrows', jend
353 write(fon, '(a,f15.3)')'xllcorner', xllcorner
354 write(fon, '(a,f15.3)')'yllcorner', yllcorner
355 write(fon, '(a,f10.3)')'cellsize_', cellsize
356 write(fon, '(a,i5)')'NODATA_value', nv
357 do j=jend,1,-1
358     write(fon, '(*(f10.3)')a(1:iend, j)
359 end do
360 close(fon)
361 !
362 end subroutine w_fasc
363 !
364 !
365 !
366 subroutine w_fasc_fmt(fon, fname, a, str, iend, jend, xllcorner, yllcorner, cellsize, nv)
367 implicit none
368 integer :: j, iend, jend, fon
369 real(8) :: xllcorner, yllcorner, cellsize, nv, a(1:iend, 1:jend)
370 character(len=100) :: fname, ciend, str
371 !
372 write(ciend,*)iend
373 open(fon, file=trim(adjustl(fname)))
374 write(fon, '(a,i5)')'ncols', iend
375 write(fon, '(a,i5)')'nrows', jend
376 write(fon, '(a,f15.3)')'xllcorner', xllcorner
377 write(fon, '(a,f15.3)')'yllcorner', yllcorner

```

```
378 write(fon, '(a,f10.3)') 'cellsize_', cellsize
379 write(fon, '(a,f10.3)') 'NODATA_value', nv
380 do j=jend,1,-1
381   write(fon, str) a(1:iend, j)
382 end do
383 close(fon)
384 !
385 end subroutine w_fasc_fmt
```

---



```

1  implicit none
2  integer, allocatable :: ibasin (:,:)
3  integer, allocatable :: i_2d (:,:), j_2d (:,:)
4  integer, allocatable :: iqx_bd_basin (:,:), iqy_bd_basin (:,:)
5  ! scalar 2d -> 1d
6  logical, allocatable :: m_cv (:,:)
7  integer :: n_ij_cv
8  integer, allocatable :: ij_cv (:,:)
9  ! vector 2d -> 1d
10 logical, allocatable :: m_u (:,:), m_v (:,:)
11 integer :: n_ij_u, n_ij_v
12 integer, allocatable :: ij_u (:,:), ij_v (:,:)
13 ! basin boundary
14 logical, allocatable :: m_u_w (:,:), m_u_e (:,:), m_v_n (:,:), m_v_s (:,:)
15 integer :: n_ij_u_w, n_ij_u_e, n_ij_v_n, n_ij_v_s, &
16           n_ij_u_we, n_ij_v_sn
17 integer, allocatable :: ij_u_w (:,:), ij_u_e (:,:), ij_v_n (:,:), ij_v_s (:,:), &
18           ij_u_we (:,:), ij_v_sn (:,:)
19 integer :: i, iend, j, jend
20 real(8) :: xllcorner, yllcorner, cellsize, dx, dy
21 character(len=100) :: fname, head, fn_bsn, fn_dir, fn_dem
22 !
23 real(8), parameter :: PI=acos(-1.d0), D2R=PI/180.d0, R2D=180.d0/PI
24 integer :: k, i1, j1
25 real(8) :: dl, deg, tele, dtr, dep
26 integer, allocatable :: idir (:,:)
27 real(8), allocatable :: z (:,:), d (:,:), grad (:,:), hsc (:,:), str (:,:)
28 character(len=100) :: ctmp, ffmt
29 !
30 open(1, file='topographyConfiguration.txt')
31 read(1,*)iend
32 read(1,*)jend
33 read(1,*)
34 read(1,*)fn_dem
35 fn_dir='output/flowDir_inRR.asc'
36 fn_bsn='output/targetArea_inRR.asc'
37 !
38 allocate(ibasin(iend, jend))
39 allocate(i_2d(iend+1, jend+1), j_2d(iend+1, jend+1))
40 allocate(iqx_bd_basin(iend+1, jend+1), iqy_bd_basin(iend+1, jend+1))
41 !
42 call r_iasc(10, fn_bsn, ibasin, iend, jend, xllcorner, yllcorner, cellsize)
43 dx=cellsize
44 dy=cellsize
45 !
46 do i=1, iend+1
47   do j=1, jend+1
48     i_2d(i, j)=i
49     j_2d(i, j)=j
50   end do
51 end do
52 ! —> set control volume
53 allocate(m_cv(iend, jend))

```

```

54 m_cv(1:iend,1:jend)=(ibasin(1:iend,1:jend)/=0)
55 n_ij_cv=count(m_cv)
56 allocate (ij_cv(n_ij_cv,2))
57 ij_cv(:,1)=pack(i_2d(1:iend,1:jend),m_cv)
58 ij_cv(:,2)=pack(j_2d(1:iend,1:jend),m_cv)
59 ! <— set control volume
60 !
61 ! —> set boundary flag
62 iqx_bd_basin(1:iend+1,1:jend+1)=1
63 iqy_bd_basin(1:iend+1,1:jend+1)=1
64 call set_flux_boundary_basin(iend,jend,ibasin(:,:), &
65     iqx_bd_basin(1:iend+1,1:jend+1),iqy_bd_basin(1:iend+1,1:jend+1))
66 ! boundary flags
67 ! 1: flux@slope qx,qy
68 ! 50: north basin qy
69 ! 51: east basin qx
70 ! 52: south basin qy
71 ! 53: west basin qx
72 ! <— set boundary flag
73
74 ! —> set flux
75 allocate(m_u(iend+1,jend+1), m_v(iend+1,jend+1))
76 !m_u(1:iend+1,1:jend+1)=(iqx_bd_basin(1:iend+1,1:jend+1)==1) ! mask
77 !m_v(1:iend+1,1:jend+1)=(iqy_bd_basin(1:iend+1,1:jend+1)==1) ! mask
78 m_u(1:iend+1,1:jend+1)=(iqx_bd_basin(1:iend+1,1:jend+1)==1 .or. &
79     iqx_bd_basin(1:iend+1,1:jend+1)==51 .or. &
80     iqx_bd_basin(1:iend+1,1:jend+1)==53) ! mask
81 m_v(1:iend+1,1:jend+1)=(iqy_bd_basin(1:iend+1,1:jend+1)==1 .or. &
82     iqy_bd_basin(1:iend+1,1:jend+1)==50 .or. &
83     iqy_bd_basin(1:iend+1,1:jend+1)==52) ! mask
84 n_ij_u=count(m_u)
85 n_ij_v=count(m_v)
86 allocate (ij_u(n_ij_u,2), ij_v(n_ij_v,2))
87 ij_u(:,1)=pack(i_2d,m_u)
88 ij_u(:,2)=pack(j_2d,m_u)
89 ij_v(:,1)=pack(i_2d,m_v)
90 ij_v(:,2)=pack(j_2d,m_v)
91 ! <— set flux
92 !
93 ! —> set boundary
94 allocate(m_u_w(iend+1,jend+1),m_u_e(iend+1,jend+1), &
95     m_v_n(iend+1,jend+1),m_v_s(iend+1,jend+1))
96 m_u_w(1:iend+1,1:jend+1)=(iqx_bd_basin(1:iend+1,1:jend+1)==53)! mask
97 m_u_e(1:iend+1,1:jend+1)=(iqx_bd_basin(1:iend+1,1:jend+1)==51)! mask
98 m_v_s(1:iend+1,1:jend+1)=(iqy_bd_basin(1:iend+1,1:jend+1)==52)! mask
99 m_v_n(1:iend+1,1:jend+1)=(iqy_bd_basin(1:iend+1,1:jend+1)==50)! mask
100 n_ij_u_w=count(m_u_w)
101 n_ij_u_e=count(m_u_e)
102 n_ij_v_s=count(m_v_s)
103 n_ij_v_n=count(m_v_n)
104 allocate (ij_u_w(n_ij_u_w,2), &
105     ij_u_e(n_ij_u_e,2), &
106     ij_v_s(n_ij_v_s,2), &
107     ij_v_n(n_ij_v_n,2))
108 ij_u_w(:,1)=pack(i_2d,m_u_w)

```

```

109 ij_u_w(:,2)=pack(j_2d,m_u_w)
110 ij_u_e(:,1)=pack(i_2d,m_u_e)
111 ij_u_e(:,2)=pack(j_2d,m_u_e)
112 ij_v_s(:,1)=pack(i_2d,m_v_s)
113 ij_v_s(:,2)=pack(j_2d,m_v_s)
114 ij_v_n(:,1)=pack(i_2d,m_v_n)
115 ij_v_n(:,2)=pack(j_2d,m_v_n)
116 n_ij_u_we=n_ij_u_w+n_ij_u_e
117 n_ij_v_sn=n_ij_v_s+n_ij_v_n
118 !
119 allocate(ij_u_we(n_ij_u_we,8),ij_v_sn(n_ij_v_sn,8))
120 !
121 do i=1,n_ij_u_w
122   !west h(i,j),(wl(i+1,j)-wl(i,j))/dx(0,0),(1,0),(0,0),flux<0
123   ij_u_we(i,1:2)=ij_u_w(i,1:2)
124   ij_u_we(i,3:8)=(/0,0,1,0,0,0/)
125 end do
126 do i=1,n_ij_u_e
127   !east h(i-1,j),(wl(i-1,j)-wl(i-2,j))/dx(-1,0),(-1,0),(-2,0),flux>0
128   ij_u_we(n_ij_u_w+i,1:2)=ij_u_e(i,1:2)
129   ij_u_we(n_ij_u_w+i,3:8)=(/-1,0,-1,0,-2,0/)
130 end do
131 do i=1,n_ij_v_s
132   !south h(i,j),(wl(i,j+1)-wl(i,j))/dy(0,0),(0,1),(0,0),flux<0
133   ij_v_sn(i,1:2)=ij_v_s(i,1:2)
134   ij_v_sn(i,3:8)=(/0,0,0,1,0,0/)
135 end do
136 do i=1,n_ij_v_n
137   !north h(i,j-1),(wl(i,j-1)-wl(i,j-2))/dy(0,-1),(0,-1),(0,-2),flux>0
138   ij_v_sn(n_ij_v_s+i,1:2)=ij_v_n(i,1:2)
139   ij_v_sn(n_ij_v_s+i,3:8)=(/0,-1,0,-1,0,-2/)
140 end do
141 ! <— set boundary
142 !
143 ! —> output
144 fname=trim(adjustl('output/controlVolume_centerPoint_inRR.txt'))
145 call point2(20,fname,xllcorner+dx*0.5,yllcorner+dy*0.5,dx,n_ij_cv,ij_cv(:,1),ij_cv(:,2))
146 fname=trim(adjustl('output/fluxPoint_x_inRR.txt'))
147 call point2(20,fname,xllcorner,yllcorner+dy*0.5,dx,n_ij_u,ij_u(:,1),ij_u(:,2))
148 fname=trim(adjustl('output/fluxPoint_y_inRR.txt'))
149 call point2(20,fname,xllcorner+dx*0.5,yllcorner,dx,n_ij_v,ij_v(:,1),ij_v(:,2))
150 !
151 fname=trim(adjustl('output/fluxPoint_xBoundary_inRR.txt'))
152 head='x,y,cvx,cvy,f1x,f1y,f2x,f2y,f3x,f3y,cs'
153 call point3(20,fname,xllcorner,yllcorner+dy*0.5,cellsize,&
154             n_ij_u_w+n_ij_u_e,ij_u_we(:,1),ij_u_we(:,2),ij_u_we(:,1:),8,head)
155 fname=trim(adjustl('output/fluxPoint_yBoundary_inRR.txt'))
156 head='x,y,cvx,cvy,f1x,f1y,f2x,f2y,f3x,f3y,cs'
157 call point3(20,fname,xllcorner+dx*0.5,yllcorner,cellsize,&
158             n_ij_v_s+n_ij_v_n,ij_v_sn(:,1),ij_v_sn(:,2),ij_v_sn(:,1:),8,head)
159 ! <— output
160 !
161 !allocate(z(iend,jend),d(iend,jend),grad(iend,jend),idir(iend,jend),hsc(iend,jend),&
162 !         str(iend,jend))
163 !call r_iasc(10,fn_dir,idir,iend,jend,xllcorner,yllcorner,cellsize)

```

```

164 !call r_fasc(10,fn_dem,z,iend,jend,xllcorner,yllcorner,cellsize)
165 !
166 !do k=1,n_ij_cv
167 !   i=ij_cv(k,1)
168 !   j=ij_cv(k,2)
169 !   call f_next(iend,jend,idir,i,j,i1,j1)
170 !   if(mod(idir(i,j),2)==0)then
171 !     dl=cellsize
172 !   else
173 !     dl=cellsize*sqrt(2.d0)
174 !   end if
175 !   deg=atan((z(i1,j1)-z(i,j))/dl)*R2D
176 !   grad(i,j)=deg
177 !end do
178 !
179 !write(ctmp,*)iend
180 !ffmt='(//trim(adjustl(ctmp))//trim(adjustl('f10.3'))//')'
181 !fname='grad.asc'
182 !call w_fasc(20,fname,ffmt,grad(:,,:),iend,jend,xllcorner,yllcorner,cellsize,-9999.d0)
183 !
184 !where(ibasin==1)
185 !   d=dep
186 !elsewhere
187 !   d=0.d0
188 !end where
189 !fname='d.asc'
190 !call w_fasc(20,fname,ffmt,d(:,,:),iend,jend,xllcorner,yllcorner,cellsize,-9999.d0)
191 !
192 !where(tele<=z)
193 !   str=dtr
194 !elsewhere
195 !   str=0.d0
196 !end where
197 !fname='str.asc'
198 !call w_fasc(20,fname,ffmt,str(:,,:),iend,jend,xllcorner,yllcorner,cellsize,-9999.d0)
199 !
200 !call cal_stability_hsc(n_ij_cv,ij_cv(:,,:),iend,jend,d,grad,hsc)
201 !write(ctmp,*)iend
202 !ffmt='(//trim(adjustl(ctmp))//trim(adjustl('f10.3'))//')'
203 !fname='hsc.asc'
204 !call w_fasc(20,fname,ffmt,hsc(:,,:),iend,jend,xllcorner,yllcorner,cellsize,-9999.d0)
205 !
206 write(*,'(a)')'---_nomal_end_---'
207 stop
208 end
209 !
210 !
211 !
212 subroutine set_flux_boundary_basin(iend,jend,ibasin,ix_bd_rs,iy_bd_rs)
213 implicit none
214 integer :: i,j,iend,jend
215 integer :: ibasin(1:iend,1:jend),ix_bd_rs(1:iend+1,1:jend+1),iy_bd_rs(1:iend+1,1:jend+1)
216 !
217 ! boundary flags
218 ! 50: north basin qy

```

```

219 ! 51: east basin qx
220 ! 52: south basin qy
221 ! 53: west basin qx
222 do i=2,iend
223   do j=2,jend
224     if(ibasin(i-1,j)==0 .and. ibasin(i,j)/=0)then
225       iqx_bd_rs(i,j)=53 !west
226     else if(ibasin(i-1,j)/=0 .and. ibasin(i,j)==0)then
227       iqx_bd_rs(i,j)=51 !east
228     else if(ibasin(i-1,j)/=0 .and. ibasin(i,j)/=0)then
229       iqx_bd_rs(i,j)=1
230     end if
231     if(ibasin(i,j-1)==0 .and. ibasin(i,j)/=0)then
232       iqy_bd_rs(i,j)=52 !south
233     else if(ibasin(i,j-1)/=0 .and. ibasin(i,j)==0)then
234       iqy_bd_rs(i,j)=50 !north
235     else if(ibasin(i,j-1)/=0 .and. ibasin(i,j)/=0)then
236       iqy_bd_rs(i,j)=1
237     end if
238   end do
239 end do
240 !
241 do i=2,iend
242   j=1
243   if(ibasin(i-1,j)==0 .and. ibasin(i,j)/=0)then
244     iqx_bd_rs(i,j)=53 !west
245   else if(ibasin(i-1,j)/=0 .and. ibasin(i,j)==0)then
246     iqx_bd_rs(i,j)=51 !east
247   end if
248 end do
249 !
250 do j=2,jend
251 i=1
252   if(ibasin(i,j-1)==0 .and. ibasin(i,j)/=0)then
253     iqy_bd_rs(i,j)=52 !south
254   else if(ibasin(i,j-1)/=0 .and. ibasin(i,j)==0)then
255     iqy_bd_rs(i,j)=50 !north
256   end if
257 end do
258 ! —> qx boundary
259 i=1
260 do j=1,jend
261   if(ibasin(i,j)/=0)then
262     iqx_bd_rs(i,j)=53 !west
263   end if
264 end do
265 !
266 i=iend+1
267 do j=1,jend
268   if(ibasin(i-1,j)/=0)then
269     iqx_bd_rs(i,j)=51 !east
270   end if
271 end do
272 ! <— qx boundary
273 !

```

```

274 ! —> qy boundary
275 j=1
276 do i=1,iend
277   if(ibasin(i,j)/=0)then
278     iqy_bd_rs(i,j)=52 !south
279   end if
280 end do
281 !
282 j=jend+1
283 do i=1,iend
284   if(ibasin(i,j-1)/=0)then
285     iqy_bd_rs(i,j)=50 !north
286   end if
287 end do
288 ! <— qy boundary
289 !
290 end subroutine set_flux_boundary_basin
291 !
292 !
293 !
294 subroutine w_fasc(fon,fname,cfmt,a,iend,jend,xllcorner,yllcorner,cellsize,nv)
295 implicit none
296 integer :: j,iend,jend, fon
297 real(8) :: xllcorner,yllcorner,cellsize,nv,a(1:iend,1:jend)
298 character(len=100) :: cfmt,fname
299 !
300 open(fon,file=trim(adjustl(fname)))
301 write(fon,'(a,i5)')'ncols',iend
302 write(fon,'(a,i5)')'nrows',jend
303 write(fon,'(a,f15.3)')'xllcorner',xllcorner
304 write(fon,'(a,f15.3)')'yllcorner',yllcorner
305 write(fon,'(a,f10.3)')'cellsize_',cellsize
306 write(fon,'(a,f10.3)')'NODATA_value',nv
307 do j=jend,1,-1
308   write(fon,trim(cfmt))a(1:iend,j)
309 end do
310 close(fon)
311 !
312 end subroutine w_fasc
313 !
314 !
315 !
316 subroutine w_iasc(fon,fname,ia,iend,jend,xllcorner,yllcorner,cellsize,nv)
317 implicit none
318 integer :: j,iend,jend, fon,nv,ia(1:iend,1:jend)
319 real(8) :: xllcorner,yllcorner,cellsize
320 character(len=100) :: fname
321 !
322 open(fon,file=trim(adjustl(fname)))
323 write(fon,'(a,i5)')'ncols',iend
324 write(fon,'(a,i5)')'nrows',jend
325 write(fon,'(a,f15.3)')'xllcorner',xllcorner
326 write(fon,'(a,f15.3)')'yllcorner',yllcorner
327 write(fon,'(a,f10.3)')'cellsize_',cellsize
328 write(fon,'(a,i5)')'NODATA_value',nv

```

```

329 do j=jend,1,-1
330   write(fon, '(*(i2))') ia(1:iend, j)
331 end do
332 close(fon)
333 !
334 end subroutine w_iasc
335 !
336 !
337 !
338 subroutine r_iasc(fon, fname, ia, iend, jend, xllcorner, yllcorner, cellsize)
339 implicit none
340 integer :: j, iend, jend, fon, ncols, nrows, ia(1:iend, 1:jend)
341 real(8) :: xllcorner, yllcorner, cellsize
342 character(len=100) :: ctmp, fname
343 !
344 open(fon, file=trim(adjustl(fname)))
345 read(fon, *) ctmp, ncols
346 read(fon, *) ctmp, nrows
347 if (ncols /= iend .or. nrows /= jend) then
348   write(*, *) 'Check_ncols_/=_iend_or_nrows_/=_jend'
349   stop
350 end if
351 read(fon, *) ctmp, xllcorner
352 read(fon, *) ctmp, yllcorner
353 read(fon, *) ctmp, cellsize
354 read(fon, *)
355 do j=jend,1,-1
356   read(fon, *) ia(1:iend, j)
357 end do
358 close(fon)
359 !
360 end subroutine r_iasc
361 !
362 !
363 !
364 subroutine r_fasc(fon, fname, a, iend, jend, xllcorner, yllcorner, cellsize)
365 implicit none
366 integer :: j, iend, jend, fon, ncols, nrows
367 real(8) :: xllcorner, yllcorner, cellsize, a(1:iend, 1:jend)
368 character(len=100) :: ctmp, fname
369 !
370 open(fon, file=trim(adjustl(fname)))
371 read(fon, *) ctmp, ncols
372 read(fon, *) ctmp, nrows
373 if (ncols /= iend .or. nrows /= jend) then
374   write(*, *) 'Check_ncols_/=_iend_or_nrows_/=_jend'
375   stop
376 end if
377 read(fon, *) ctmp, xllcorner
378 read(fon, *) ctmp, yllcorner
379 read(fon, *) ctmp, cellsize
380 read(fon, *)
381 do j=jend,1,-1
382   read(fon, *) a(1:iend, j)
383 end do

```

```

384 close(fon)
385 !
386 end subroutine r_fasc
387 !
388 !
389 !
390 subroutine point(fon,fname,a,iend,jend,xllcorner,yllcorner,cellsize)
391 implicit none
392 integer :: i,j,iend,jend,fon,a(1:iend,1:jend),iflg
393 real(8) :: xllcorner,yllcorner,cellsize,x,y
394 character(len=100) :: fname
395 !
396 open(fon,file=trim(adjustl(fname)))
397 write(fon,'(a)')'x,y,i,j,flag'
398 do i=1,iend
399   do j=1,jend
400     x=xllcorner+cellsize*float(i-1)
401     y=yllcorner+cellsize*float(j-1)
402     iflg=int(a(i,j))
403     write(fon,'(2(f15.3,a),3(i5,a))')x,',',y,',',i,',',j,',',iflg
404   end do
405 end do
406 close(fon)
407 !
408 end subroutine point
409 !
410 !
411 !
412 subroutine point2(fon,fname,xllcorner,yllcorner,cellsize,n_1d,i_1d,j_1d)
413 implicit none
414 integer :: ni,n_1d,i_1d(1:n_1d),j_1d(1:n_1d),i,j,fon
415 real(8) :: xllcorner,yllcorner,cellsize
416 character(len=100) :: fname
417 !
418 open(fon,file=trim(adjustl(fname)))
419 write(fon,*)n_1d
420 write(fon,'(a)')'x,y,i,j,cs'
421 do ni=1,n_1d
422   i=i_1d(ni)
423   j=j_1d(ni)
424   write(fon,'(2(f15.3,a),2(i5,a),f10.3)')&
425     xllcorner+cellsize*float(i-1),',',yllcorner+cellsize*float(j-1),',',i,',',j,',',cellsize
426 end do
427 close(fon)
428 !
429 end subroutine point2
430 !
431 !
432 !
433 subroutine point3(fon,fname,xllcorner,yllcorner,cellsize,n_1d,i_1d,j_1d,flg,n_flg,head)
434 implicit none
435 integer :: ni,n_1d,n_flg,i_1d(n_1d),j_1d(n_1d),flg(n_1d,n_flg),i,j,fon
436 real(8) :: xllcorner,yllcorner,cellsize
437 character(len=100) :: head,fname,cn_flg
438 !

```



```

439 open(fon, file=trim(adjustl(fname)))
440 write(fon,*)n_1d
441 write(fon,'(a)')head
442 write(cn_flg,*)n_flg
443 do ni=1,n_1d
444   i=i_1d(ni)
445   j=j_1d(ni)
446   write(fon,'(2(f15.3,:',","),'//trim(adjustl(cn_flg))//'(i5,:',","),f10.3)') &
447   xllcorner+cellsize*float(i-1),yllcorner+cellsize*float(j-1),flg(ni,:), cellsize
448 end do
449 close(fon)
450 !
451 end subroutine point3
452 !
453 !
454 !
455 subroutine f_next(iend,jend,idir,i,j,i1,j1)
456 implicit none
457 integer :: i,j,iend,jend,i1,j1,idir(iend,jend)
458 !
459 ! nw n ne ! 3 2 1
460 ! \ | / ! \ | /
461 ! w-+- e ! 4-+- 8
462 ! / | \ ! / | \
463 ! sw s se ! 5 6 7
464 if( idir(i,j)==1)then
465   i1=i+1
466   j1=j+1
467 else if(idir(i,j)==2)then
468   i1=i
469   j1=j+1
470 else if(idir(i,j)==3)then
471   i1=i-1
472   j1=j+1
473 else if(idir(i,j)==4)then
474   i1=i-1
475   j1=j
476 else if(idir(i,j)==5)then
477   i1=i-1
478   j1=j-1
479 else if(idir(i,j)==6)then
480   i1=i
481   j1=j-1
482 else if(idir(i,j)==7)then
483   i1=i+1
484   j1=j-1
485 else if(idir(i,j)==8)then
486   i1=i+1
487   j1=j
488 else if(i1<1 .or. i1>iend .or. j1<1 .or. j1>jend)then
489   i1=-1
490   j1=-1
491 else
492   write(*,'(a,4i5)')'river_course?_@_i,j,dir,iriv=',i,j,idir(i,j)
493   write(*,*)'Don't come here?'

```

```

494  stop
495  end if
496  !
497  end subroutine f_next
498  !
499  !
500  !
501  subroutine cal_stability_hsc(n_1d, ij_1d, iend, jend, d, grad, hsc)
502  implicit none
503  real(8), parameter :: PI=acos(-1.d0), D2R=PI/180.d0, R2D=180.d0/PI
504  real(8), parameter :: g=9.8d0, sig=2650.d0, rho=1000.d0, lamda=0.4d0, pw=0.1d0, &
505  c=3000.d0, tanp=tan(35.d0*D2R)
506  integer :: ni, n_1d, ij_1d(n_1d, 2), i, j, iend, jend
507  real(8) :: d(iend, jend), grad(iend, jend), c2, hsc(iend, jend), hsc0, csta, tant, cost
508  !
509  csta=1.d0-lamda
510  do ni=1, n_1d
511  i=ij_1d(ni, 1)
512  j=ij_1d(ni, 2)
513  tant=grad(i, j)*D2R
514  cost=cos(atan(tant))
515  c2=c/(rho*g*d(i, j)*cost*tanp)
516  hsc0=((1.d0-tant/tanp)*(csta*sig/rho+pw)+c2) / &
517  ((1.d0-tant/tanp)*(csta+pw)+tant/tanp)
518  hsc(i, j)=hsc0/d(i, j)
519  end do
520  !
521  end subroutine cal_stability_hsc

```

---

```

1 implicit none
2 real(8), parameter :: PI=acos(-1.d0), deg2rad=PI/180.d0, rad2deg=180.d0/PI
3 integer,parameter :: upcell=1 ! start point of river
4 integer,allocatable :: ibasin(:, :), idrain(:, :), iriver(:, :), istream_order(:, :), iacc(:, :), &
5     i_2d(:, :), j_2d(:, :), i1_2d(:, :), j1_2d(:, :), i2_2d(:, :), j2_2d(:, :))
6 integer :: iend, jend, i, j, k, i1, j1, i2, j2, itmp, iacc_thresh, nirs, itr, iso_max
7 integer,allocatable :: i_1d(:, :), j_1d(:, :), in_1d(:, :), jn_1d(:, :), ip_1d(:, :), jp_1d(:, :), idrain_1d(:, :),
8     iso_1d(:, :), iacc_1d(:, :), &
9     iriver_sta_i(:, :), iriver_sta_j(:, :))
9 integer :: n_1d, n_linf, iflg_in0, iflg_in1
10 real(8), allocatable :: x_2d(:, :), y_2d(:, :), z_2d(:, :), ! ,acc(iend, jend)
11 real(8), allocatable :: x_1d(:, :), y_1d(:, :), z_1d(:, :), ! ,acc_1d(:)
12 real(8) :: x, y, xllcorner, yllcorner, cellsize, ba
13 character(len=100) :: fname, ctmp, ftmp
14 !
15 integer, allocatable :: id_from(:, :), id_to(:, :), idx(:, :))
16 integer :: iexit
17 logical :: mask
18 real(8), allocatable :: b_1d(:, :), d_1d(:, :), dx(:, :))
19 real(8) :: dx0, dy0, x_vec, y_vec, deg, dl, grad, wm, wp, wmin, dm, dp, dmin, mn, dsoil
20 !
21 open(1, file='topographyConfiguration.txt')
22 read(1, *) iend
23 read(1, *) jend
24 allocate( ibasin(iend, jend), idrain(iend, jend), iriver(iend, jend), istream_order(iend, jend), iacc
25     (iend, jend), &
26     i_2d(iend, jend), j_2d(iend, jend), i1_2d(iend, jend), j1_2d(iend, jend), i2_2d(iend, jend),
27     j2_2d(iend, jend), &
28     x_2d(iend, jend), y_2d(iend, jend), z_2d(iend, jend))
29 read(1, *)
30 read(1, *)
31 call r_fasc(10, fname, z_2d, iend, jend, xllcorner, yllcorner, cellsize)
32 !
33 read(1, *)
34 read(1, *)
35 read(1, *)
36 read(1, *)
37 read(1, *)
38 read(1, *) iacc_thresh
39 !
40 fname='output/targetArea_inRR.asc'
41 call r_iasc(10, fname, ibasin, iend, jend, xllcorner, yllcorner, cellsize)
42 !
43 fname='output/flowAcc_inRR.asc'
44 call r_iasc(10, fname, iacc, iend, jend, xllcorner, yllcorner, cellsize)
45 where(iacc(1:iend, 1:jend)>iacc_thresh .and. ibasin(1:iend, 1:jend)/=0)
46     iriver(1:iend, 1:jend)=1
47 elsewhere
48     iriver(1:iend, 1:jend)=0
49 end where
50 !

```

```

51 fname='output/flowDir_inRR.asc'
52 call r_iasc(10,fname, idrain , iend , jend , xllcorner , yllcorner , cellsize)
53 !
54 ! -> set start of river
55 ! nw n ne ! 3 2 1
56 ! \ | / ! \ | /
57 ! w - e ! 4 - 8
58 ! / | \ ! / | \
59 ! sw s se ! 5 6 7
60 nirs=0
61 open(20, file='misc/river_bounds.txt')
62 write(20,*) 'x,y,i,j'
63 do i=1,iend
64   do j=1,jend
65     if(iriver(i,j)==1)then
66       !
67       iflg_in1=0
68       if(idrain(i, j+1)==6 .and. iriver(i, j+1)==1)iflg_in1=iflg_in1+1 ! n
69       if(idrain(i+1,j+1)==5 .and. iriver(i+1,j+1)==1)iflg_in1=iflg_in1+1 ! ne
70       if(idrain(i+1, j)==4 .and. iriver(i+1, j)==1)iflg_in1=iflg_in1+1 ! e
71       if(idrain(i+1,j-1)==3 .and. iriver(i+1,j-1)==1)iflg_in1=iflg_in1+1 ! se
72       if(idrain(i, j-1)==2 .and. iriver(i, j-1)==1)iflg_in1=iflg_in1+1 ! s
73       if(idrain(i-1,j-1)==1 .and. iriver(i-1,j-1)==1)iflg_in1=iflg_in1+1 ! sw
74       if(idrain(i-1, j)==8 .and. iriver(i-1, j)==1)iflg_in1=iflg_in1+1 ! w
75       if(idrain(i-1,j+1)==7 .and. iriver(i-1,j+1)==1)iflg_in1=iflg_in1+1 ! nw
76       !
77       iflg_in0=0
78       if(idrain(i, j+1)==6 .and. iriver(i, j+1)==0)iflg_in0=iflg_in0+1 ! n
79       if(idrain(i+1,j+1)==5 .and. iriver(i+1,j+1)==0)iflg_in0=iflg_in0+1 ! ne
80       if(idrain(i+1, j)==4 .and. iriver(i+1, j)==0)iflg_in0=iflg_in0+1 ! e
81       if(idrain(i+1,j-1)==3 .and. iriver(i+1,j-1)==0)iflg_in0=iflg_in0+1 ! se
82       if(idrain(i, j-1)==2 .and. iriver(i, j-1)==0)iflg_in0=iflg_in0+1 ! s
83       if(idrain(i-1,j-1)==1 .and. iriver(i-1,j-1)==0)iflg_in0=iflg_in0+1 ! sw
84       if(idrain(i-1, j)==8 .and. iriver(i-1, j)==0)iflg_in0=iflg_in0+1 ! w
85       if(idrain(i-1,j+1)==7 .and. iriver(i-1,j+1)==0)iflg_in0=iflg_in0+1 ! nw
86       !
87       itmp=0
88       if(iriver(i, j+1)==1)itmp=itmp+1 ! n
89       if(iriver(i+1,j+1)==1)itmp=itmp+1 ! ne
90       if(iriver(i+1, j)==1)itmp=itmp+1 ! e
91       if(iriver(i+1,j-1)==1)itmp=itmp+1 ! se
92       if(iriver(i, j-1)==1)itmp=itmp+1 ! s
93       if(iriver(i-1,j-1)==1)itmp=itmp+1 ! sw
94       if(iriver(i-1, j)==1)itmp=itmp+1 ! w
95       if(iriver(i-1,j+1)==1)itmp=itmp+1 ! nw
96       !
97       if(iflg_in0 >= upcell .and. iflg_in1==0)then
98         !if(itmp==1 .or. iflg_in==0)then
99           call f_next(iend , jend , idrain , i , j , i1 , j1)
100          if (iriver(i1 , j1)==1)then
101            nirs=nirs+1
102            ! write(*,*) 'inlet(i,j)=' , i , j
103            write(20, '(2(f15.3,a),3(i5,a))') xllcorner+cellsize*(0.5+i-1),',', &
104            yllcorner+cellsize*(0.5+j-1),',', i ,',', j
105          else

```

```

106         ! write(*,*)'outlet',i,j,'Ent.'
107         !read(*,*)
108     end if
109 end if
110 end if
111 end do
112 end do
113 close(20)
114 !
115 allocate(iriver_sta_i(nirs),iriver_sta_j(nirs))
116 !write(*,*)'Input OK, Ent.', 'No. point =',nirs
117 ! ← set start of river
118 !
119 istream_order(:,:)=0
120 x_2d(:,:)= -1
121 y_2d(:,:)= -1
122 i_2d(:,:)=0
123 j_2d(:,:)=0
124 i1_2d(:,:)=0
125 j1_2d(:,:)=0
126 i2_2d(:,:)=0
127 j2_2d(:,:)=0
128 open(20,file='misc/river_bounds.txt')
129 read(20,*)
130 do k=1,nirs
131     read(20,*)ftmp,ftmp,iriver_sta_i(k),iriver_sta_j(k) !,nirs_dirflag(k)
132     i=iriver_sta_i(k)
133     j=iriver_sta_j(k)
134     !write(ctmp,'(i3.3)')k
135     !write(ctmp,'(a,i3.3,a,i3.3)')'i',i,'_j',j
136     !fname='rivers/river_'//trim(adjustl(ctmp))//'.csv'
137     !open(100,file=fname)
138     !write(100,'(a)')'x,y,z,i,j,i-1,j-1,i+1,j+1'
139     !write(*,*)iriver_sta_i(k),iriver_sta_j(k) !,nirs_dirflag(k)
140     itmp=0
141     !i=iriver_sta_i(k)
142     !j=iriver_sta_j(k)
143     istream_order(i,j)=1
144     itr=1
145     i2=0
146     j2=0
147     do while(itmp==0)
148         call f_next(iend,jend,idrain,i,j,i1,j1)
149         x=xllcorner+cellsize*(0.5+i-1)
150         y=yllcorner+cellsize*(0.5+j-1)
151         x_2d(i,j)=x
152         y_2d(i,j)=y
153         i_2d(i,j)=i
154         j_2d(i,j)=j
155         i1_2d(i,j)=i1
156         j1_2d(i,j)=j1
157         i2_2d(i,j)=i2
158         j2_2d(i,j)=j2
159         !if(iriver(i1,j1)==0)then
160         ! write(100,'(3(f15.3,a),6(i5,a))')x,',',y,',',z_2d(i,j),',',i,',',j,',',i2,',',j2

```

```

, , , 0, , , 0
161 !else
162 ! write(100,'(3(f15.3,a),6(i5,a))'x, , , y, , , z_2d(i,j), , , i, , , j, , , i2, , , j2, , ,
i1, , , j1
163 !end if
164 if(i1==1 .or. j1==1 .or. ibasin(i1,j1)==0)then
165     exit
166 end if
167 !
168 iso_max=0
169 n_inf=0
170 if(idrain(i1+1,j1+1)==5 .and. idrain(i,j)/=5) iso_max=max(iso_max,istream_order(i1+1,j1
+1))
171 if(idrain(i1 ,j1+1)==6 .and. idrain(i,j)/=6) iso_max=max(iso_max,istream_order(i1 ,j1
+1))
172 if(idrain(i1-1,j1+1)==7 .and. idrain(i,j)/=7) iso_max=max(iso_max,istream_order(i1-1,j1
+1))
173 if(idrain(i1-1,j1 )==8 .and. idrain(i,j)/=8) iso_max=max(iso_max,istream_order(i1-1,j1
))
174 if(idrain(i1-1,j1-1)==1 .and. idrain(i,j)/=1) iso_max=max(iso_max,istream_order(i1-1,j1
-1))
175 if(idrain(i1 ,j1-1)==2 .and. idrain(i,j)/=2) iso_max=max(iso_max,istream_order(i1 ,j1
-1))
176 if(idrain(i1+1,j1-1)==3 .and. idrain(i,j)/=3) iso_max=max(iso_max,istream_order(i1+1,j1
-1))
177 if(idrain(i1+1,j1 )==4 .and. idrain(i,j)/=4) iso_max=max(iso_max,istream_order(i1+1,j1
))
178 !if (iso_max>0) write(*,*)'iso_max',iso_max
179 if(istream_order(i,j)<iso_max)then
180     istream_order(i1,j1)=iso_max
181 else if(istream_order(i,j)==iso_max)then
182     istream_order(i1,j1)=iso_max+1
183 else if(istream_order(i,j)>iso_max)then
184     istream_order(i1,j1)=istream_order(i,j)
185 else
186     write(*,*)'from',istream_order(i,j)
187     write(*,*)'to',istream_order(i1,j1)
188 end if
189 i2=i
190 j2=j
191 i=i1
192 j=j1
193 !write(*,*)'iteration =',itr
194 ! itr=itr+1
195 end do
196 !close(100)
197 end do
198 close(20,status='delete')
199 !
200 fname='misc/streamOrder.asc'
201 call w_iasc(20,fname,istream_order(:,:),iend,jend,xllcorner,yllcorner,cellsize,0)
202 !fname='IO/iriver.asc'
203 !call w_iasc(20,fname,iriver(:,:),iend,jend,xllcorner,yllcorner,cellsize,0)
204 n_1d=count(istream_order(:,)/=0)
205 !write(*,*)'n_1d=',n_1d

```

```

206 allocate(i_1d(n_1d),j_1d(n_1d),in_1d(n_1d),jn_1d(n_1d),ip_1d(n_1d),jp_1d(n_1d),idrain_1d(
      n_1d),iso_1d(n_1d),iacc_1d(n_1d))
207 allocate(x_1d(n_1d),y_1d(n_1d),z_1d(n_1d))!,acc_1d(n_1d))
208 i_1d(1:n_1d)=pack(i_2d(:, :), istream_order(:, :)/=0)
209 j_1d(1:n_1d)=pack(j_2d(:, :), istream_order(:, :)/=0)
210 in_1d(1:n_1d)=pack(i1_2d(:, :), istream_order(:, :)/=0)
211 jn_1d(1:n_1d)=pack(j1_2d(:, :), istream_order(:, :)/=0)
212 ip_1d(1:n_1d)=pack(i2_2d(:, :), istream_order(:, :)/=0)
213 jp_1d(1:n_1d)=pack(j2_2d(:, :), istream_order(:, :)/=0)
214 !
215 iacc_1d(1:n_1d)=pack(iacc(:, :), istream_order(:, :)/=0)
216 idrain_1d(1:n_1d)=pack(idrain(:, :), istream_order(:, :)/=0)
217 iso_1d(1:n_1d)=pack(istream_order(:, :), istream_order(:, :)/=0)
218 !
219 x_1d(1:n_1d)=pack(x_2d(:, :), istream_order(:, :)/=0)
220 y_1d(1:n_1d)=pack(y_2d(:, :), istream_order(:, :)/=0)
221 z_1d(1:n_1d)=pack(z_2d(:, :), istream_order(:, :)/=0)
222 !open(20, file='IO/river_struct.txt')
223 !write(20, '(a)') 'x,y,z,i,j,i-1,j-1,i+1,j+1,km2,iacc,drain,stream_order'
224 !do i=1,n_1d
225 ! !write(*, '(2f15.3,6i5)') x_1d(i),y_1d(i),i_1d(i),j_1d(i),in_1d(i),jn_1d(i),ip_1d(i),jp_1d(
      i)
226 ! ba=cellsize*cellsize*float(iacc_1d(i))/1000000.d0
227 ! if(i_1d(i)==0 .or. j_1d(i)==0)then
228 !   write(*, '(a,10i5)') i=0 or j=0 @', i_1d(i),j_1d(i),ip_1d(i),jp_1d(i),in_1d(i),jn_1d(i)
229 ! end if
230 ! write(20, '(3(f15.3,a),6(i5,a),(f10.5,a),(i10,a),2(i5,a))') x_1d(i),',',y_1d(i),',',z_1d(i)
      ',',',i_1d(i),',',j_1d(i),',', &
231 ! ip_1d(i),',',jp_1d(i),',',in_1d(i),',',jn_1d(i),',',ba,',',iacc_1d(i),',',idrain_1d(i)
      ',',',iso_1d(i)
232 !end do
233 !close(20)
234 allocate(id_from(n_1d,3),id_to(n_1d),idx(3),dx(n_1d),b_1d(n_1d),d_1d(n_1d))!,acc_1d(n_1d))
235 ! Width & depth Tanaka & Sayama 2018 DPR1
236 ! Width
237 wm=4.73d0
238 wp=0.58d0
239 wmin=5.d0
240 ! Depth
241 dm=1.57d0
242 dp=0.33d0
243 dmin=1.d0
244 !
245 mn=0.1d0 ! Manning coefficient
246 dsoil=1.d0
247 !
248 do i=1,n_1d
249   id_from(i,:)=0
250   itr=0
251   do j=1,n_1d
252     if(in_1d(j)==i_1d(i) .and. jn_1d(j)==j_1d(i))then
253       itr=itr+1
254       id_from(i,itr)=j
255     end if
256   end do

```

```

257  idx(1:itr)=id_from(i,1:itr)
258  iexit=0
259  do j=1,itr
260      mask=iso_1d(idx(j))==maxval(iso_1d(idx(1:itr)))
261      if(mask .and. iacc_1d(idx(j))==maxval(iacc_1d(idx(1:itr))) .or. &
262         mask .and. count(iso_1d(idx(1:itr))==maxval(iso_1d(idx(1:itr))))==1)then
263         id_from(i,1:itr)=cshift(idx(1:itr),j-1)
264         exit
265     end if
266 end do
267 end do
268 !
269 do i=1,n_1d
270     idx(:)=0
271     iexit=0
272     do j=1,n_1d
273         do k=1,3
274             if(id_from(j,k)==i .and. minval(id_from(j,1:k))>=1)then
275                 idx(1)=j
276                 iexit=1
277                 exit
278             end if
279         end do
280     end do
281     if(iexit==1)exit
282 !
283 do j=1,n_1d
284     if(ip_1d(j)==i_1d(i) .and. jp_1d(j)==j_1d(i))then
285         idx(2)=j
286     end if
287 end do
288 !
289 id_to(i)=maxval(idx(:))
290 if(id_to(i)==0)then
291     dx(i)=-999
292 else
293     dx0=x_1d(id_to(i))-x_1d(i)
294     dy0=y_1d(id_to(i))-y_1d(i)
295     dx(i)=(dx0*dx0+dy0*dy0)**0.5d0
296 end if
297 ba=cellsize*cellsize*float(iacc_1d(i))/1000000.d0
298 b_1d(i)=max(wmin,wm*ba**wp)
299 d_1d(i)=max(dmin,dm*ba**dp)
300 end do
301 !
302 open(20,file='output/streamConfiguration_inRR.txt')
303 write(20,'(i5,a)')n_1d,'#_of_node'
304 !write(20,'(a)')'x,y,z,b,dep,dx,id,id_to,id_from1,id_from2,id_from3,iacc,i,j'
305 write(20,'(a)')'id,id_to,id_from1,id_from2,id_from3,z_2d,z_1d,zs_1d,dx,b,n,iacc,x_2d,y_2d,
    i_2d,j_2d'
306 do i=1,n_1d
307     if(id_to(i)==0)then
308         if(mod(idrain_1d(i)*(-45.d0)+90.d0,90.d0)==0)then
309             !dx(i)=1.d0
310             dx(i)=maxval(dx)/2.d0**0.5d0

```





```

364         dl=cellsize*2.d0**0.5d0
365     end if
366     x=xllcorner+cellsize*(0.5+i-1)
367     y=yllcorner+cellsize*(0.5+j-1)
368     deg=idrain(i,j)*(45.d0)
369     x=x+dl*0.5d0*cos(deg2rad*deg)
370     y=y+dl*0.5d0*sin(deg2rad*deg)
371     deg=idrain(i,j)*(-45.d0)+90.d0
372     write(20,'(2(f15.3,'',''),2(i5,'',''),2(f10.3,'',''))')x,y,i,j,dl,deg
373 end if
374 end do
375 end do
376 close(20)
377 !
378 write(*,'(a)')'---_nomal_end_---'
379 stop
380 end
381 !
382 !
383 !
384 subroutine f_next(iend,jend,idrain,i,j,i1,j1)
385 implicit none
386 integer :: i,j,iend,jend,i1,j1
387 integer :: idrain(iend,jend)
388 !
389 ! mw n ne ! 3 2 1
390 ! \ | / ! \ | /
391 ! w-+-e ! 4-+-8
392 ! / | \ ! / | \
393 ! sw s se ! 5 6 7
394 !idrain(i,j)=abs(idrain(i,j))
395 if( idrain(i,j)==1)then
396     i1=i+1
397     j1=j+1
398 else if(idrain(i,j)==2)then
399     i1=i
400     j1=j+1
401 else if(idrain(i,j)==3)then
402     i1=i-1
403     j1=j+1
404 else if(idrain(i,j)==4)then
405     i1=i-1
406     j1=j
407 else if(idrain(i,j)==5)then
408     i1=i-1
409     j1=j-1
410 else if(idrain(i,j)==6)then
411     i1=i
412     j1=j-1
413 else if(idrain(i,j)==7)then
414     i1=i+1
415     j1=j-1
416 else if(idrain(i,j)==8)then
417     i1=i+1
418     j1=j

```

```

419 !else if(idrain(i,j)<=-1 .or. iriver(i,j)==0)then
420 ! write(*,'(a,4i5)')'river end @ i,j,dir,iriv=',i,j,idrain(i,j),iriver(i,j)
421 ! itmp=1
422 else if(i1<1 .or. i1>iend .or. j1<1 .or. j1>jend)then
423 ! write(*,'(a,4i5)')'river end @ i,j,dir,iriv=',i,j,idrain(i,j)
424   i1=-1
425   j1=-1
426   !itmp=1
427 else
428   write(*,'(a,4i5)')'river_course?_@_i,j,dir,iriv=',i,j,idrain(i,j) !,iriver(i,j)
429   write(*,'*')'Don't_come_here?'
430   stop
431 end if
432 !
433 end subroutine f_next
434 !
435 !
436 !
437 subroutine r_iasc(fon,fname,ia,iend,jend,xllcorner,yllcorner,cellsize)
438 implicit none
439 integer :: j,iend,jend,fon,ncols,nrows,ia(1:iend,1:jend)
440 real(8) :: xllcorner,yllcorner,cellsize
441 character(len=100) :: ctmp,fname
442 !
443 open(fon,file=trim(adjustl(fname)))
444 read(fon,*)ctmp,ncols
445 read(fon,*)ctmp,nrows
446 if (ncols /= iend .or. nrows /= jend) then
447   write(*,'*')'Check_ncols_/_iend_or_nrows_/_jend'
448   stop
449 end if
450 read(fon,*)ctmp,xllcorner
451 read(fon,*)ctmp,yllcorner
452 read(fon,*)ctmp,cellsize
453 read(fon,*)
454 do j=jend,1,-1
455   read(fon,*)ia(1:iend,j)
456 end do
457 close(fon)
458 !
459 end subroutine r_iasc
460 !
461 !
462 !
463 subroutine r_fasc(fon,fname,a,iend,jend,xllcorner,yllcorner,cellsize)
464 implicit none
465 integer :: j,iend,jend,fon,ncols,nrows
466 real(8) :: xllcorner,yllcorner,cellsize,a(1:iend,1:jend)
467 character(len=100) :: ctmp,fname
468 !
469 open(fon,file=trim(adjustl(fname)))
470 read(fon,*)ctmp,ncols
471 read(fon,*)ctmp,nrows
472 if (ncols /= iend .or. nrows /= jend) then
473   write(*,'*')'Check_ncols_/_iend_or_nrows_/_jend'

```

```

474  stop
475  end if
476  read(fon,*)ctmp,xllcorner
477  read(fon,*)ctmp,yllcorner
478  read(fon,*)ctmp,cellsize
479  read(fon,*)
480  do j=jend,1,-1
481    read(fon,*)a(1:iend,j)
482  end do
483  close(fon)
484  !
485  end subroutine r_fasc
486  !
487  !
488  !
489  subroutine w_iasc(fon,fname,ia,iend,jend,xllcorner,yllcorner,cellsize,nv)
490  implicit none
491  integer :: j,iend,jend,fon,nv,ia(1:iend,1:jend)
492  real(8) :: xllcorner,yllcorner,cellsize
493  character(len=100) :: fname
494  !
495  open(fon,file=trim(adjustl(fname)))
496  write(fon,'(a,i5)')'ncols',iend
497  write(fon,'(a,i5)')'nrows',jend
498  write(fon,'(a,f15.3)')'xllcorner',xllcorner
499  write(fon,'(a,f15.3)')'yllcorner',yllcorner
500  write(fon,'(a,f10.3)')'cellsize_',cellsize
501  write(fon,'(a,i5)')'NODATA_value',nv
502  do j=jend,1,-1
503    write(fon,'(*i2)')ia(1:iend,j)
504  end do
505  close(fon)
506  !
507  end subroutine w_iasc

```

---

## 2.2 02\_DF

01\_mk\_floodplainConfigurationFiles.f90

```

1  implicit none
2  integer, allocatable :: ibasin (:,:)
3  integer, allocatable :: i_2d (:,:), j_2d (:,:)
4  integer, allocatable :: iqx_bd_basin (:,:), iqy_bd_basin (:,:)
5  ! scalar 2d -> 1d
6  logical, allocatable :: m_cv (:,:)
7  integer :: n_ij_cv
8  integer, allocatable :: ij_cv (:,:)
9  ! vector 2d -> 1d
10 logical, allocatable :: m_u (:,:), m_v (:,:)
11 integer :: n_ij_u, n_ij_v
12 integer, allocatable :: ij_u (:,:), ij_v (:,:)
13 ! basin boundary
14 logical, allocatable :: m_u_w (:,:), m_u_e (:,:), m_v_n (:,:), m_v_s (:,:)
15 integer :: n_ij_u_w, n_ij_u_e, n_ij_v_n, n_ij_v_s, &
16           n_ij_u_we, n_ij_v_sn
17 integer, allocatable :: ij_u_w (:,:), ij_u_e (:,:), ij_v_n (:,:), ij_v_s (:,:), &
18           ij_u_we (:,:), ij_v_sn (:,:)
19 integer :: i, iend, j, jend
20 real(8) :: xllcorner, yllcorner, cellsize, dx, dy
21 character(len=100) :: fname, head, fn_bsn, fn_dir, fn_dem
22 !
23 real(8), parameter :: PI=acos(-1.d0), D2R=PI/180.d0, R2D=180.d0/PI
24 integer :: k, i1, j1
25 real(8) :: dl, deg, tele, dtr, dep
26 integer, allocatable :: idir (:,:)
27 real(8), allocatable :: z (:,:), d (:,:), grad (:,:), hsc (:,:), str (:,:)
28 character(len=100) :: ctmp, i_fmt, f_fmt
29 !
30 open(1, file='topographyConfiguration.txt')
31 read(1,*) iend
32 read(1,*) jend
33 read(1,*)
34 read(1,*) fn_bsn
35 read(1,*) fn_dem
36 read(1,*) fn_dir
37 close(1)
38 !
39 allocate( ibasin(iend, jend), z(iend, jend), idir(iend, jend))
40 allocate( i_2d(iend+1, jend+1), j_2d(iend+1, jend+1))
41 allocate( iqx_bd_basin(iend+1, jend+1), iqy_bd_basin(iend+1, jend+1))
42 !
43 call r_iasc(10, fn_bsn, ibasin, iend, jend, xllcorner, yllcorner, cellsize)
44 fname='output/targetArea_inDF.asc'
45 write(ctmp,*) iend
46 write(f_fmt,*) len(trim(adjustl(ctmp)))+1
47 f_fmt='*(i'//trim(adjustl(f_fmt))//')'
48 ! call w_iasc(20, fname, iriver (:,:), iend, jend, xllcorner, yllcorner, cellsize, 0)
49 call w_iasc_fmt(20, fname, ibasin, f_fmt, iend, jend, xllcorner, yllcorner, cellsize, 0)
50 !

```

```

51 call r_fasc(10,fn_dem,z,iend,jend,xllcorner,yllcorner,cellsize)
52 fname='output/elevation_inDF.asc'
53 f_fmt='*(f10.3))'
54 call w_fasc_fmt(20,fname,z,f_fmt,iend,jend,xllcorner,yllcorner,cellsize,-999.d0)
55 !
56 call r_iasc(10,fn_dir,idir,iend,jend,xllcorner,yllcorner,cellsize)
57 fname='output/flowDir_inDF.asc'
58 write(ctmp,*)maxval(idir)
59 write(i_fmt,*)len(trim(adjustl(ctmp)))+1+1
60 i_fmt='*(i'//trim(adjustl(i_fmt))//')'
61 call w_iasc_fmt(20,fname,idir,i_fmt,iend,jend,xllcorner,yllcorner,cellsize,0)
62 !
63 dx=cellsize
64 dy=cellsize
65 !
66 do i=1,iend+1
67   do j=1,jend+1
68     i_2d(i,j)=i
69     j_2d(i,j)=j
70   end do
71 end do
72 ! —> set control volume
73 allocate(m_cv(iend,jend))
74 m_cv(1:iend,1:jend)=(ibasin(1:iend,1:jend)/=0)
75 n_ij_cv=count(m_cv)
76 allocate(ij_cv(n_ij_cv,2))
77 ij_cv(:,1)=pack(i_2d(1:iend,1:jend),m_cv)
78 ij_cv(:,2)=pack(j_2d(1:iend,1:jend),m_cv)
79 ! <— set control volume
80 !
81 ! —> set boundary flag
82 iqx_bd_basin(1:iend+1,1:jend+1)=1
83 iqy_bd_basin(1:iend+1,1:jend+1)=1
84 call set_flux_boundary_basin(iend,jend,ibasin(:, :), &
85   iqx_bd_basin(1:iend+1,1:jend+1),iqy_bd_basin(1:iend+1,1:jend+1))
86 ! boundary flags
87 ! 1: flux@slope qx,qy
88 ! 50: north basin qy
89 ! 51: east basin qx
90 ! 52: south basin qy
91 ! 53: west basin qx
92 ! <— set boundary flag
93
94 ! —> set flux
95 allocate(m_u(iend+1,jend+1), m_v(iend+1,jend+1))
96 !m_u(1:iend+1,1:jend+1)=(iqx_bd_basin(1:iend+1,1:jend+1)==1) ! mask
97 !m_v(1:iend+1,1:jend+1)=(iqy_bd_basin(1:iend+1,1:jend+1)==1) ! mask
98 m_u(1:iend+1,1:jend+1)=(iqx_bd_basin(1:iend+1,1:jend+1)==1 .or. &
99   iqx_bd_basin(1:iend+1,1:jend+1)==51 .or. &
100  iqx_bd_basin(1:iend+1,1:jend+1)==53) ! mask
101 m_v(1:iend+1,1:jend+1)=(iqy_bd_basin(1:iend+1,1:jend+1)==1 .or. &
102  iqy_bd_basin(1:iend+1,1:jend+1)==50 .or. &
103  iqy_bd_basin(1:iend+1,1:jend+1)==52) ! mask
104 n_ij_u=count(m_u)
105 n_ij_v=count(m_v)

```

```

106 allocate( ij_u( n_ij_u , 2) , ij_v( n_ij_v , 2) )
107 ij_u( :, 1) = pack( i_2d , m_u )
108 ij_u( :, 2) = pack( j_2d , m_u )
109 ij_v( :, 1) = pack( i_2d , m_v )
110 ij_v( :, 2) = pack( j_2d , m_v )
111 ! ← set flux
112 !
113 ! → set boundary
114 allocate( m_u_w( iend+1, jend+1) , m_u_e( iend+1, jend+1) , &
115           m_v_n( iend+1, jend+1) , m_v_s( iend+1, jend+1) )
116 m_u_w( 1: iend+1, 1: jend+1) = ( iqx_bd_basin( 1: iend+1, 1: jend+1) == 53 ) ! mask
117 m_u_e( 1: iend+1, 1: jend+1) = ( iqx_bd_basin( 1: iend+1, 1: jend+1) == 51 ) ! mask
118 m_v_s( 1: iend+1, 1: jend+1) = ( ixy_bd_basin( 1: iend+1, 1: jend+1) == 52 ) ! mask
119 m_v_n( 1: iend+1, 1: jend+1) = ( ixy_bd_basin( 1: iend+1, 1: jend+1) == 50 ) ! mask
120 n_ij_u_w = count( m_u_w )
121 n_ij_u_e = count( m_u_e )
122 n_ij_v_s = count( m_v_s )
123 n_ij_v_n = count( m_v_n )
124 allocate( ij_u_w( n_ij_u_w , 2) , &
125           ij_u_e( n_ij_u_e , 2) , &
126           ij_v_s( n_ij_v_s , 2) , &
127           ij_v_n( n_ij_v_n , 2) )
128 ij_u_w( :, 1) = pack( i_2d , m_u_w )
129 ij_u_w( :, 2) = pack( j_2d , m_u_w )
130 ij_u_e( :, 1) = pack( i_2d , m_u_e )
131 ij_u_e( :, 2) = pack( j_2d , m_u_e )
132 ij_v_s( :, 1) = pack( i_2d , m_v_s )
133 ij_v_s( :, 2) = pack( j_2d , m_v_s )
134 ij_v_n( :, 1) = pack( i_2d , m_v_n )
135 ij_v_n( :, 2) = pack( j_2d , m_v_n )
136 n_ij_u_we = n_ij_u_w + n_ij_u_e
137 n_ij_v_sn = n_ij_v_s + n_ij_v_n
138 !
139 allocate( ij_u_we( n_ij_u_we , 8) , ij_v_sn( n_ij_v_sn , 8) )
140 !
141 do i = 1, n_ij_u_w
142   ! west h( i , j ) , ( wl( i+1 , j ) - wl( i , j ) ) / dx ( 0 , 0 ) , ( 1 , 0 ) , ( 0 , 0 ) , flux < 0
143   ij_u_we( i , 1: 2) = ij_u_w( i , 1: 2)
144   ij_u_we( i , 3: 8) = ( / 0 , 0 , 1 , 0 , 0 , 0 / )
145 end do
146 do i = 1, n_ij_u_e
147   ! east h( i-1 , j ) , ( wl( i-1 , j ) - wl( i-2 , j ) ) / dx ( -1 , 0 ) , ( -1 , 0 ) , ( -2 , 0 ) , flux > 0
148   ij_u_we( n_ij_u_w + i , 1: 2) = ij_u_e( i , 1: 2)
149   ij_u_we( n_ij_u_w + i , 3: 8) = ( / -1 , 0 , -1 , 0 , -2 , 0 / )
150 end do
151 do i = 1, n_ij_v_s
152   ! south h( i , j ) , ( wl( i , j+1 ) - wl( i , j ) ) / dy ( 0 , 0 ) , ( 0 , 1 ) , ( 0 , 0 ) , flux < 0
153   ij_v_sn( i , 1: 2) = ij_v_s( i , 1: 2)
154   ij_v_sn( i , 3: 8) = ( / 0 , 0 , 0 , 1 , 0 , 0 / )
155 end do
156 do i = 1, n_ij_v_n
157   ! north h( i , j-1 ) , ( wl( i , j-1 ) - wl( i , j-2 ) ) / dy ( 0 , -1 ) , ( 0 , -1 ) , ( 0 , -2 ) , flux > 0
158   ij_v_sn( n_ij_v_s + i , 1: 2) = ij_v_n( i , 1: 2)
159   ij_v_sn( n_ij_v_s + i , 3: 8) = ( / 0 , -1 , 0 , -1 , 0 , -2 / )
160 end do

```

```

161 ! <— set boundary
162 !
163 ! —> output
164 fname=trim(adjustl('output/controlVolume_centerPoint_inDF.txt'))
165 call point2(20,fname,xllcorner+dx*0.5,yllcorner+dy*0.5,dx,n_ij_cv,ij_cv(:,1),ij_cv(:,2))
166 fname=trim(adjustl('output/fluxPoint_x_inDF.txt'))
167 call point2(20,fname,xllcorner,yllcorner+dy*0.5,dx,n_ij_u,ij_u(:,1),ij_u(:,2))
168 fname=trim(adjustl('output/fluxPoint_y_inDF.txt'))
169 call point2(20,fname,xllcorner+dx*0.5,yllcorner,dx,n_ij_v,ij_v(:,1),ij_v(:,2))
170 !
171 fname=trim(adjustl('output/fluxPoint_xBoundary_inDF.txt'))
172 head='x,y,cvx,cvy,f1x,f1y,f2x,f2y,f3x,f3y,cs'
173 call point3(20,fname,xllcorner,yllcorner+dy*0.5,cellsize,&
174             n_ij_u_w+n_ij_u_e,ij_u_we(:,1),ij_u_we(:,2),ij_u_we(:,1:),8,head)
175 fname=trim(adjustl('output/fluxPoint_yBoundary_inDF.txt'))
176 head='x,y,cvx,cvy,f1x,f1y,f2x,f2y,f3x,f3y,cs'
177 call point3(20,fname,xllcorner+dx*0.5,yllcorner,cellsize,&
178             n_ij_v_s+n_ij_v_n,ij_v_sn(:,1),ij_v_sn(:,2),ij_v_sn(:,1:),8,head)
179 ! <— output
180 !
181 open(10,file='output/fluxPoint_wBoundary_inDF.txt')
182 write(10,*)n_ij_u_w
183 write(10,'(a)')'x,y,i,j'
184 do k=1,n_ij_u_w
185     i=ij_u_w(k,1)
186     j=ij_u_w(k,2)
187     write(10,'(2(f15.3,:","),2(i5,:","))') &
188         xllcorner+cellsize*float(i-1),yllcorner+dy*0.5+cellsize*float(j-1),i,j
189 end do
190 close(10)
191 !
192 open(10,file='output/fluxPoint_eBoundary_inDF.txt')
193 write(10,*)n_ij_u_e
194 write(10,'(a)')'x,y,i,j'
195 do k=1,n_ij_u_e
196     i=ij_u_e(k,1)
197     j=ij_u_e(k,2)
198     write(10,'(2(f15.3,:","),2(i5,:","))') &
199         xllcorner+cellsize*float(i-1),yllcorner+dy*0.5+cellsize*float(j-1),i,j
200 end do
201 close(10)
202 !
203 open(10,file='output/fluxPoint_sBoundary_inDF.txt')
204 write(10,*)n_ij_v_s
205 write(10,'(a)')'x,y,i,j'
206 do k=1,n_ij_v_s
207     i=ij_v_s(k,1)
208     j=ij_v_s(k,2)
209     write(10,'(2(f15.3,:","),2(i5,:","))') &
210         xllcorner+dx*0.5+cellsize*float(i-1),yllcorner+cellsize*float(j-1),i,j
211 end do
212 close(10)
213 !
214 open(10,file='output/fluxPoint_nBoundary_inDF.txt')
215 write(10,*)n_ij_v_n

```



```

216 write(10,'(a)')'x,y,i,j'
217 do k=1,n_ij_v_n
218   i=ij_v_n(k,1)
219   j=ij_v_n(k,2)
220   write(10,'(2(f15.3,:","),2(i5,:","))') &
221     xllcorner+dx*0.5+cellsize*float(i-1),yllcorner+cellsize*float(j-1),i,j
222 end do
223 close(10)
224 !
225 do i=1,n_ij_u_e
226   !east h(i-1, j),(wl(i-1, j)-wl(i-2, j))/dx (-1, 0),(-1, 0),(-2, 0),flux>0
227   ij_u_we(n_ij_u_w+i,1:2)=ij_u_e(i,1:2)
228   ij_u_we(n_ij_u_w+i,3:8)=(/-1,0,-1,0,-2,0/)
229 end do
230 do i=1,n_ij_v_s
231   !south h(i, j),(wl(i, j+1)-wl(i, j))/dy (0, 0),(0, 1),(0, 0),flux<0
232   ij_v_sn(i,1:2)=ij_v_s(i,1:2)
233   ij_v_sn(i,3:8)=(/0,0,0,1,0,0/)
234 end do
235 do i=1,n_ij_v_n
236   !north h(i, j-1),(wl(i, j-1)-wl(i, j-2))/dy (0, -1),(0, -1),(0, -2),flux>0
237   ij_v_sn(n_ij_v_s+i,1:2)=ij_v_n(i,1:2)
238   ij_v_sn(n_ij_v_s+i,3:8)=(/0,-1,0,-1,0,-2/)
239 end do
240 !
241 write(*,'(a)')'---_nomal_end_---'
242 stop
243 end
244 !
245 !
246 !
247 subroutine set_flux_boundary_basin(iend,jend,ibasin,ix_bd_rs,iqy_bd_rs)
248 implicit none
249 integer :: i,j,iend,jend
250 integer :: ibasin(1:iend,1:jend),ix_bd_rs(1:iend+1,1:jend+1),iqy_bd_rs(1:iend+1,1:jend+1)
251 !
252 ! boundary flags
253 ! 50: north basin qy
254 ! 51: east basin qx
255 ! 52: south basin qy
256 ! 53: west basin qx
257 do i=2,iend
258   do j=2,jend
259     if(ibasin(i-1,j)==0 .and. ibasin(i,j)/=0)then
260       ix_bd_rs(i,j)=53 !west
261     else if(ibasin(i-1,j)/=0 .and. ibasin(i,j)==0)then
262       ix_bd_rs(i,j)=51 !east
263     else if(ibasin(i-1,j)/=0 .and. ibasin(i,j)/=0)then
264       ix_bd_rs(i,j)=1
265     end if
266     if(ibasin(i,j-1)==0 .and. ibasin(i,j)/=0)then
267       iqy_bd_rs(i,j)=52 !south
268     else if(ibasin(i,j-1)/=0 .and. ibasin(i,j)==0)then
269       iqy_bd_rs(i,j)=50 !north
270     else if(ibasin(i,j-1)/=0 .and. ibasin(i,j)/=0)then

```

```

271     iqy_bd_rs(i,j)=1
272     end if
273 end do
274 end do
275 !
276 do i=2,iend
277     j=1
278     if(ibasin(i-1,j)==0 .and. ibasin(i,j)/=0)then
279         iqx_bd_rs(i,j)=53 !west
280     else if(ibasin(i-1,j)/=0 .and. ibasin(i,j)==0)then
281         iqx_bd_rs(i,j)=51 !east
282     end if
283 end do
284 !
285 do j=2,jend
286     i=1
287     if(ibasin(i,j-1)==0 .and. ibasin(i,j)/=0)then
288         iqy_bd_rs(i,j)=52 !south
289     else if(ibasin(i,j-1)/=0 .and. ibasin(i,j)==0)then
290         iqy_bd_rs(i,j)=50 !north
291     end if
292 end do
293 ! —> qx boundary
294 i=1
295 do j=1,jend
296     if(ibasin(i,j)/=0)then
297         iqx_bd_rs(i,j)=53 !west
298     end if
299 end do
300 !
301 i=iend+1
302 do j=1,jend
303     if(ibasin(i-1,j)/=0)then
304         iqx_bd_rs(i,j)=51 !east
305     end if
306 end do
307 ! <— qx boundary
308 !
309 ! —> qy boundary
310 j=1
311 do i=1,iend
312     if(ibasin(i,j)/=0)then
313         iqy_bd_rs(i,j)=52 !south
314     end if
315 end do
316 !
317 j=jend+1
318 do i=1,iend
319     if(ibasin(i,j-1)/=0)then
320         iqy_bd_rs(i,j)=50 !north
321     end if
322 end do
323 ! <— qy boundary
324 !
325 end subroutine set_flux_boundary_basin

```

```

326 !
327 !
328 !
329 subroutine w_fasc(fon , fname , cfmt , a , iend , jend , xllcorner , yllcorner , cellsize , nv)
330 implicit none
331 integer :: j , iend , jend , fon
332 real(8) :: xllcorner , yllcorner , cellsize , nv , a(1:iend , 1:jend)
333 character(len=100) :: cfmt , fname
334 !
335 open(fon , file=trim(adjustl(fname)))
336 write(fon , '(a,i5)') 'ncols' , iend
337 write(fon , '(a,i5)') 'nrows' , jend
338 write(fon , '(a,f15.3)') 'xllcorner' , xllcorner
339 write(fon , '(a,f15.3)') 'yllcorner' , yllcorner
340 write(fon , '(a,f10.3)') 'cellsize' , cellsize
341 write(fon , '(a,f10.3)') 'NODATA_value' , nv
342 do j=jend,1,-1
343   write(fon , trim(cfmt))a(1:iend , j)
344 end do
345 close(fon)
346 !
347 end subroutine w_fasc
348 !
349 !
350 !
351 subroutine w_iasc(fon , fname , ia , iend , jend , xllcorner , yllcorner , cellsize , nv)
352 implicit none
353 integer :: j , iend , jend , fon , nv , ia(1:iend , 1:jend)
354 real(8) :: xllcorner , yllcorner , cellsize
355 character(len=100) :: fname
356 !
357 open(fon , file=trim(adjustl(fname)))
358 write(fon , '(a,i5)') 'ncols' , iend
359 write(fon , '(a,i5)') 'nrows' , jend
360 write(fon , '(a,f15.3)') 'xllcorner' , xllcorner
361 write(fon , '(a,f15.3)') 'yllcorner' , yllcorner
362 write(fon , '(a,f10.3)') 'cellsize' , cellsize
363 write(fon , '(a,i5)') 'NODATA_value' , nv
364 do j=jend,1,-1
365   write(fon , '(*(i2)') ia(1:iend , j)
366 end do
367 close(fon)
368 !
369 end subroutine w_iasc
370 !
371 !
372 !
373 subroutine r_iasc(fon , fname , ia , iend , jend , xllcorner , yllcorner , cellsize)
374 implicit none
375 integer :: j , iend , jend , fon , ncols , nrows , ia(1:iend , 1:jend)
376 real(8) :: xllcorner , yllcorner , cellsize
377 character(len=100) :: ctmp , fname
378 !
379 open(fon , file=trim(adjustl(fname)))
380 read(fon , *)ctmp , ncols

```

```

381 read(fon,*)ctmp,nrows
382 if (ncols /= iend .or. nrows /= jend) then
383   write(*,*)'Check_ncols_/_iend_or_nrows_/_jend'
384   stop
385 end if
386 read(fon,*)ctmp,xllcorner
387 read(fon,*)ctmp,yllcorner
388 read(fon,*)ctmp,cellsize
389 read(fon,*)
390 do j=jend,1,-1
391   read(fon,*)ia(1:iend,j)
392 end do
393 close(fon)
394 !
395 end subroutine r_iasc
396 !
397 !
398 !
399 subroutine r_fasc(fon,fname,a,iend,jend,xllcorner,yllcorner,cellsize)
400 implicit none
401 integer :: j,iend,jend, fon,ncols,nrows
402 real(8) :: xllcorner,yllcorner,cellsize,a(1:iend,1:jend)
403 character(len=100) :: ctmp,fname
404 !
405 open(fon,file=trim(adjustl(fname)))
406 read(fon,*)ctmp,ncols
407 read(fon,*)ctmp,nrows
408 if (ncols /= iend .or. nrows /= jend) then
409   write(*,*)'Check_ncols_/_iend_or_nrows_/_jend'
410   stop
411 end if
412 read(fon,*)ctmp,xllcorner
413 read(fon,*)ctmp,yllcorner
414 read(fon,*)ctmp,cellsize
415 read(fon,*)
416 do j=jend,1,-1
417   read(fon,*)a(1:iend,j)
418 end do
419 close(fon)
420 !
421 end subroutine r_fasc
422 !
423 !
424 !
425 subroutine point(fon,fname,a,iend,jend,xllcorner,yllcorner,cellsize)
426 implicit none
427 integer :: i,j,iend,jend, fon,a(1:iend,1:jend),iflg
428 real(8) :: xllcorner,yllcorner,cellsize,x,y
429 character(len=100) :: fname
430 !
431 open(fon,file=trim(adjustl(fname)))
432 write(fon,'(a)')'x,y,i,j,flag'
433 do i=1,iend
434   do j=1,jend
435     x=xllcorner+cellsize*float(i-1)

```

```

436     y=yllcorner+cellsize*float(j-1)
437     iflg=int(a(i,j))
438     write(fon,'(2(f15.3,a),3(i5,a))')x,',',y,',',i,',',j,',',iflg
439 end do
440 end do
441 close(fon)
442 !
443 end subroutine point
444 !
445 !
446 !
447 subroutine point2(fon,fname,xllcorner,yllcorner,cellsize,n_1d,i_1d,j_1d)
448 implicit none
449 integer :: ni,n_1d,i_1d(1:n_1d),j_1d(1:n_1d),i,j,fon
450 real(8) :: xllcorner,yllcorner,cellsize
451 character(len=100) :: fname
452 !
453 open(fon,file=trim(adjustl(fname)))
454 write(fon,*)n_1d
455 write(fon,'(a)')'x,y,i,j,cs'
456 do ni=1,n_1d
457   i=i_1d(ni)
458   j=j_1d(ni)
459   write(fon,'(2(f15.3,a),2(i5,a),f10.3)')&
460     xllcorner+cellsize*float(i-1),',',yllcorner+cellsize*float(j-1),',',i,',',j,',',cellsize
461 end do
462 close(fon)
463 !
464 end subroutine point2
465 !
466 !
467 !
468 subroutine point3(fon,fname,xllcorner,yllcorner,cellsize,n_1d,i_1d,j_1d,flg,n_flg,head)
469 implicit none
470 integer :: ni,n_1d,n_flg,i_1d(n_1d),j_1d(n_1d),flg(n_1d,n_flg),i,j,fon
471 real(8) :: xllcorner,yllcorner,cellsize
472 character(len=100) :: head,fname,cn_flg
473 !
474 open(fon,file=trim(adjustl(fname)))
475 write(fon,*)n_1d
476 write(fon,'(a)')head
477 write(cn_flg,*)n_flg
478 do ni=1,n_1d
479   i=i_1d(ni)
480   j=j_1d(ni)
481   write(fon,'(2(f15.3,:',"',')//trim(adjustl(cn_flg))//'(i5,:',"',')f10.3)') &
482     xllcorner+cellsize*float(i-1),yllcorner+cellsize*float(j-1),flg(ni,:),cellsize
483 end do
484 close(fon)
485 !
486 end subroutine point3
487 !
488 !
489 !
490 subroutine f_next(iend,jend,idir,i,j,i1,j1)

```

```

491 implicit none
492 integer :: i,j,iend,jend,i1,j1,idir(iend,jend)
493 !
494 ! nw n ne ! 3 2 1
495 ! \ | / ! \ | /
496 ! w-+- e ! 4-+- 8
497 ! / | \ ! / | \
498 ! sw s se ! 5 6 7
499 if( idir(i,j)==1)then
500   i1=i+1
501   j1=j+1
502 else if( idir(i,j)==2)then
503   i1=i
504   j1=j+1
505 else if( idir(i,j)==3)then
506   i1=i-1
507   j1=j+1
508 else if( idir(i,j)==4)then
509   i1=i-1
510   j1=j
511 else if( idir(i,j)==5)then
512   i1=i-1
513   j1=j-1
514 else if( idir(i,j)==6)then
515   i1=i
516   j1=j-1
517 else if( idir(i,j)==7)then
518   i1=i+1
519   j1=j-1
520 else if( idir(i,j)==8)then
521   i1=i+1
522   j1=j
523 else if(i1<1 .or. i1>iend .or. j1<1 .or. j1>jend)then
524   i1=-1
525   j1=-1
526 else
527   write(*, '(a,4i5)')'river_course?_@_i,j,dir,iriv=',i,j, idir(i,j)
528   write(*,*)'Don't_come_here?'
529   stop
530 end if
531 !
532 end subroutine f_next
533 !
534 !
535 !
536 subroutine cal_stability_hsc(n_1d, ij_1d, iend, jend, d, grad, hsc)
537 implicit none
538 real(8), parameter :: PI=acos(-1.d0), D2R=PI/180.d0, R2D=180.d0/PI
539 real(8), parameter :: g=9.8d0, sig=2650.d0, rho=1000.d0, lamda=0.4d0, pw=0.1d0, &
540   c=3000.d0, tanp=tan(35.d0*D2R)
541 integer :: ni, n_1d, ij_1d(n_1d,2), i, j, iend, jend
542 real(8) :: d(iend, jend), grad(iend, jend), c2, hsc(iend, jend), hsc0, csta, tant, cost
543 !
544 csta=1.d0-lamda
545 do ni=1,n_1d

```

```

546  i:=ij_1d(ni,1)
547  j:=ij_1d(ni,2)
548  tant=grad(i,j)*D2R
549  cost=cos(atan(tant))
550  c2=c/(rho*g*d(i,j)*cost*tanp)
551  hsc0=((1.d0-tant/tanp)*(csta*sig/rho+pw)+c2) / &
552      ((1.d0-tant/tanp)*(csta+pw)+tant/tanp)
553  hsc(i,j)=hsc0/d(i,j)
554  end do
555  !
556  end subroutine cal_stability_hsc
557  !
558  !
559  !
560  subroutine w_iasc_fmt(fon,fname,ia, str, iend, jend, xllcorner, yllcorner, cellsize, nv)
561  implicit none
562  integer :: j, iend, jend, fon, nv, ia(1:iend,1:jend)
563  real(8) :: xllcorner, yllcorner, cellsize
564  character(len=100) :: fname, ciend, str
565  !
566  write(ciend,*)iend
567  open(fon, file=trim(adjustl(fname)))
568  write(fon, '(a,i5)')'ncols', iend
569  write(fon, '(a,i5)')'nrows', jend
570  write(fon, '(a,f15.3)')'xllcorner', xllcorner
571  write(fon, '(a,f15.3)')'yllcorner', yllcorner
572  write(fon, '(a,f10.3)')'cellsize_□', cellsize
573  write(fon, '(a,i5)')'NODATA_value', nv
574  do j=jend,1,-1
575     write(fon, str)ia(1:iend, j)
576  end do
577  close(fon)
578  !
579  end subroutine w_iasc_fmt
580  !
581  !
582  !
583  subroutine w_fasc_fmt(fon,fname,a, str, iend, jend, xllcorner, yllcorner, cellsize, nv)
584  implicit none
585  integer :: j, iend, jend, fon
586  real(8) :: xllcorner, yllcorner, cellsize, nv, a(1:iend,1:jend)
587  character(len=100) :: fname, ciend, str
588  !
589  write(ciend,*)iend
590  open(fon, file=trim(adjustl(fname)))
591  write(fon, '(a,i5)')'ncols', iend
592  write(fon, '(a,i5)')'nrows', jend
593  write(fon, '(a,f15.3)')'xllcorner', xllcorner
594  write(fon, '(a,f15.3)')'yllcorner', yllcorner
595  write(fon, '(a,f10.3)')'cellsize_□', cellsize
596  write(fon, '(a,f10.3)')'NODATA_value', nv
597  do j=jend,1,-1
598     write(fon, str)a(1:iend, j)
599  end do
600  close(fon)

```

```
601 !  
602 end subroutine w_fasc_fmt
```

---



```

1 implicit none
2 real(8), parameter :: PI=acos(-1.d0), deg2rad=PI/180.d0, rad2deg=180.d0/PI
3 !
4 integer :: n_ij_cv, n_ij_u, n_ij_v
5 integer, allocatable :: ij_cv(:, :), ij_u(:, :), ij_v(:, :), ibasin(:, :)
6 !
7 integer, allocatable :: ij_1d2d(:, :, :), i_to(:), i_from(:)
8 real(8), allocatable :: xy_1d2d(:, :, :),
9 real(8), allocatable :: f2d(:, :, :),
10 !
11 integer :: iend_1d, iofs, jofs, itr
12 integer :: i, j, iend, jend, itmp, k
13 real(8) :: x, y, xllcorner, yllcorner, cellsize, x1, x2, y1, y2
14 real(8) :: dist, dlx, dly, tmp, cs_c
15 character(len=100) :: fn, ctmp, ffmt
16 !
17 fn='output/targetArea_inDF.asc'
18 open(10, file=fn)
19 read(10, *) ctmp, iend
20 read(10, *) ctmp, jend
21 close(10)
22 write(*, *) iend, jend
23 allocate(ibasin(iend, jend))
24 call r_iasc(10, fn, ibasin, iend, jend, xllcorner, yllcorner, cellsize)
25 !
26 open(1, file='output/controlVolume_centerPoint_inDF.txt')
27 read(1, *) n_ij_cv
28 allocate(ij_cv(n_ij_cv, 2))
29 read(1, *)
30 do k=1, n_ij_cv
31   read(1, *) x, y, ij_cv(k, 1:2)
32 end do
33 close(1)
34 !
35 open(1, file='output/fluxPoint_x_inDF.txt')
36 read(1, *) n_ij_u
37 allocate(ij_u(n_ij_u, 2))
38 read(1, *)
39 do k=1, n_ij_u
40   read(1, *) x, y, ij_u(k, 1:2)
41 end do
42 close(1)
43 !
44 open(1, file='output/fluxPoint_y_inDF.txt')
45 read(1, *) n_ij_v
46 allocate(ij_v(n_ij_v, 2))
47 read(1, *)
48 do k=1, n_ij_v
49   read(1, *) x, y, ij_v(k, 1:2)
50 end do
51 close(1)
52 !
53 open(10, file='input/streamConfiguration_inRR.txt')

```

```

54 read(10,*)iend_1d
55 allocate(ij_1d2d(iend_1d,2,1),xy_1d2d(iend_1d,2),i_to(iend_1d),i_from(iend_1d))
56 read(10,*)
57 do i=1,iend_1d
58   read(10,*)itmp,i_to(i),i_from(i),itmp,itmp,tmp,tmp,tmp,dist,tmp,tmp,itmp,&
59     xy_1d2d(i,1),xy_1d2d(i,2),ij_1d2d(i,1,1),ij_1d2d(i,2,1)
60   if(i==1)then
61     cs_c=dist
62   else if(dist < cs_c)then
63     cs_c=dist
64   end if
65   !write(*,*)xy_1d2d(i,1),xy_1d2d(i,2),ij_1d2d(i,1),ij_1d2d(i,2),i_to(i)
66 end do
67 deallocate(ij_1d2d)
68 close(10)
69 !
70 itr=int(cs_c/cellsize)
71 allocate(ij_1d2d(iend_1d,2,itr))
72 ij_1d2d(:, :, :)=0
73 write(*,*)'#_of_div.=',itr
74 open(20,file='output/streamFloodplainConnection.txt')
75 write(20,'(i5,a)')itr,' '#_of_div'
76 write(20,'(a)')'x,y,i2d,j2d,i1d,itr,dist'
77 do i=1,iend_1d
78   !dist=(cellsize*db1e(iend)*cellsize*db1e(jend))*0.5d0
79   do k=1,n_ij_cv
80     x=xl1corner+cellsize*(db1e(ij_cv(k,1)-1)+0.5d0)
81     y=yl1corner+cellsize*(db1e(ij_cv(k,2)-1)+0.5d0)
82     dlx=xy_1d2d(i,1)-x
83     dly=xy_1d2d(i,2)-y
84     if(k==1)dist=(dlx*dlx+dly*dly)**0.5d0
85     if((dlx*dlx+dly*dly)**0.5d0 <= dist)then
86       ij_1d2d(i,1,1)=ij_cv(k,1)
87       ij_1d2d(i,2,1)=ij_cv(k,2)
88       dist=(dlx*dlx+dly*dly)**0.5d0
89     end if
90   end do
91   if(ij_1d2d(i,1,1)==0 .or. ij_1d2d(i,2,1)==0)then
92     write(*,*)'vol_1d_>_2d_point_missing_@1D(i)',i,'<<<STOP>>>'
93     stop
94   end if
95   !
96   x1=xy_1d2d(i,1)!x1
97   y1=xy_1d2d(i,2)!y1
98   x2=xy_1d2d(i_to(i),1)!x2
99   y2=xy_1d2d(i_to(i),2)!y2
100  !
101  if(i_to(i)==0)then
102    x1=xy_1d2d(i_from(i),1)!x1
103    y1=xy_1d2d(i_from(i),2)!y1
104    x2=xy_1d2d(i,1)!x2
105    y2=xy_1d2d(i,2)!y2
106  end if
107  !
108  if(x2==x1)then

```

```

109     iofs=0
110     else
111         iofs=nint(sign(1.d0,x2-x1))
112     end if
113     if(y2=y1)then
114         jofs=0
115     else
116         jofs=nint(sign(1.d0,y2-y1))
117     end if
118     !
119     do j=1,itr
120         if(j==1)then
121             else
122                 ij_1d2d(i,1,j)=ij_1d2d(i,1,j-1)+iofs
123                 ij_1d2d(i,2,j)=ij_1d2d(i,2,j-1)+jofs
124             end if
125             if(dist<cellsize/2.d0**0.5d0)then
126                 if(ibasins(ij_1d2d(i,1,j),ij_1d2d(i,2,j))==0)then
127                     write(*,*)'outof_basin?_@2D(i,j)',ij_1d2d(i,1,j),ij_1d2d(i,2,j)!,'<<<STOP>>>'
128                     !stop
129                 else
130                     write(20,'(2(f15.3,a),2(i5,a),(i10,a),(i5,a),f10.3)')&
131                         xllcorner+cellsize*(dble(ij_1d2d(i,1,j)-1)+0.5d0),',', &
132                         yllcorner+cellsize*(dble(ij_1d2d(i,2,j)-1)+0.5d0),',', &
133                         ij_1d2d(i,1,j),',',ij_1d2d(i,2,j),',',i,',',j,',',dist
134                     end if
135                 else
136                     !write(*,*)'dist=',dist,'<<<STOP>>>'
137                     !stop
138                 end if
139             end do
140         end do
141     close(20)
142     !
143     allocate(f2d(iend,jend))
144     f2d=0.d0
145     fn='output/d.asc'
146     write(ctmp,*)maxval(f2d)
147     write(ffmt,*)len(trim(adjustl(ctmp)))+1
148     ffmt='*(f'//trim(adjustl(ffmt))//')'
149     ffmt='*(f10.3)'
150     write(*,*)trim(adjustl(fn)),',',ffmt
151     call w_fasc_fmt(20,fn,f2d,ffmt,iend,jend,xllcorner,yllcorner,cellsize,0.d0)
152     !
153     f2d=0.d0
154     fn='output/hs_ini.asc'
155     write(ctmp,*)maxval(f2d)
156     write(ffmt,*)len(trim(adjustl(ctmp)))+1
157     ffmt='*(f'//trim(adjustl(ffmt))//')'
158     ffmt='*(f10.3)'
159     write(*,*)trim(adjustl(fn)),',',ffmt
160     call w_fasc_fmt(20,fn,f2d,ffmt,iend,jend,xllcorner,yllcorner,cellsize,0.d0)
161     !
162     write(*, '(a)')'---_nomal_end_---'
163     stop

```

```

164 end
165 !
166 !
167 !
168 subroutine r_iasc(fon , fname , ia , iend , jend , xllcorner , yllcorner , cellsize )
169 implicit none
170 integer :: j , iend , jend , fon , ncols , nrows , ia (1:iend ,1:jend )
171 real(8) :: xllcorner , yllcorner , cellsize
172 character(len=100) :: ctmp , fname
173 !
174 open(fon , file=trim(adjustl(fname)))
175 read(fon ,*)ctmp , ncols
176 read(fon ,*)ctmp , nrows
177 if (ncols /= iend .or. nrows /= jend) then
178   write(* ,*)'Check_ncols_/=_iend_or_nrows_/=_jend'
179   stop
180 end if
181 read(fon ,*)ctmp , xllcorner
182 read(fon ,*)ctmp , yllcorner
183 read(fon ,*)ctmp , cellsize
184 read(fon ,*)
185 do j=jend,1,-1
186   read(fon ,*) ia (1:iend , j)
187 end do
188 close(fon)
189 !
190 end subroutine r_iasc
191 !
192 !
193 !
194 subroutine w_fasc_fmt(fon , fname , a , str , iend , jend , xllcorner , yllcorner , cellsize , nv)
195 implicit none
196 integer :: j , iend , jend , fon
197 real(8) :: xllcorner , yllcorner , cellsize , nv , a(1:iend ,1:jend )
198 character(len=100) :: fname , ciend , str
199 !
200 write(ciend ,*) iend
201 open(fon , file=trim(adjustl(fname)))
202 write(fon , '(a,i5)') 'ncols' , iend
203 write(fon , '(a,i5)') 'nrows' , jend
204 write(fon , '(a,f15.3)') 'xllcorner' , xllcorner
205 write(fon , '(a,f15.3)') 'yllcorner' , yllcorner
206 write(fon , '(a,f10.3)') 'cellsize_' , cellsize
207 write(fon , '(a,f10.3)') 'NODATA_value' , nv
208 do j=jend,1,-1
209   write(fon , str) a(1:iend , j)
210 end do
211 close(fon)
212 !
213 end subroutine w_fasc_fmt

```

---

### 3 03\_MK\_RAIN

01\_mk\_rain.py

```
1 import zipfile
2 import io
3 import numpy as np
4 import sys
5 import datetime
6 import os
7 import glob
8 import matplotlib.pyplot as plt
9 #
10 def aoi(rasterfn):
11     #https://pcjericks.github.io/py-gdalogr-cookbook/raster_layers.html#get-
12     raster-band
13     #import gdal
14     from osgeo import gdal
15     raster = gdal.Open(rasterfn)
16     geotransform = raster.GetGeoTransform()
17     originX = geotransform[0]
18     originY = geotransform[3]
19     pixelWidth = geotransform[1]
20     pixelHeight = geotransform[5]
21     band = raster.GetRasterBand(1)
22     array = band.ReadAsArray()
23     cols = array.shape[1]
24     rows = array.shape[0]
25     xmin, xmax=originX, originX+pixelWidth*cols
26     ymin, ymax=originY+pixelHeight*rows, originY
27     return(xmin, xmax, ymin, ymax)
28 #
29 def eva(doy):
30     import math
31     e0, m, ofs, lam=2.6, 1.7, 205, 365
32     e=math.cos((doy-ofs)/lam*math.pi*2.)*m+e0
33     e=e/24.
34 #
35 #
36 #
37 #
38 def trs(lon, lat, src_EPSG, dst_EPSG):
39     from osgeo import ogr, osr, gdal
40     src_srs, dst_srs = osr.SpatialReference(), osr.SpatialReference()
41     src_srs.ImportFromEPSG(src_EPSG)
42     dst_srs.ImportFromEPSG(dst_EPSG)
43     trans = osr.CoordinateTransformation(src_srs, dst_srs)
44     return trans.TransformPoint(lat, lon)[::-1][1:3] # lat, lon -> lon, lat
45 #
46 ROI_xmin, ROI_xmax, ROI_ymin, ROI_ymax=aoi('targetArea_inRR.asc')
47 print('ROI(xy)', ROI_xmin, ROI_xmax, ROI_ymin, ROI_ymax)
48 pnts = [(ROI_xmin, ROI_ymin), (ROI_xmax, ROI_ymax)]
49 pnts=[trs(*pnt, 6670, 6668) for pnt in pnts]
```

```

50 [(ROI_xmin, ROI_ymin), (ROI_xmax, ROI_ymax)]=pnts
51 print('ROI(II)',ROI_xmin,ROI_xmax,ROI_ymin,ROI_ymax)
52 files=glob.glob(os.path.join('XRAIN_from_DIAS','201707040000-201707080000-10-
    FUK-130.7031-33.5000-130.8500-33.3688.zip'))
53 files.sort()
54 for fn in files:
55     fn_ps='_'.join(os.path.basename(fn).split('-')[0:2])
56     myzip=zipfile.ZipFile(fn,mode='r')
57     fs=', '
58     ls='\n'
59     dt_min=float(os.path.basename(fn).split('-')[2])
60     print('dt=',dt_min)
61     xmin,ymin=[float(fn[:-4].split('-')[i]) for i in [4,7]]
62     xmax,ymax=[float(fn[:-4].split('-')[i]) for i in [6,5]]
63     fname=myzip.namelist()[0]
64     myfile=myzip.open(fname)
65     zipcont = io.StringIO(myfile.read().decode())
66     output=np.loadtxt(zipcont,dtype=np.float,delimiter=fs)
67     jend,iend=np.shape(output)
68     dx,dy=(xmax-xmin)/float(iend-1),(ymax-ymin)/float(jend-1)
69     xmin=xmin+0.5*dx
70     ymin=ymin+0.5*dy
71     print('AOI(II)',xmin,xmin+float(iend)*dx,ymin,ymin+dy*float(jend))
72     #print(dx,dy)
73     x=np.linspace(xmin,xmin+(iend-1)*dx,iend)
74     y=np.linspace(ymin,ymin+(jend-1)*dy,jend)
75     X,Y=np.meshgrid(x,y)
76     ij=np.where((ROI_xmin-dx < X) & (X < ROI_xmax+dx) & \
77                (ROI_ymin-dy < Y) & (Y < ROI_ymax+dy))
78     data=(np.vstack((X[ij],Y[ij]))).T
79     xy=np.array([trs(*pnt,6668,6670) for pnt in data])
80     fo=open('grid.csv','w')
81     fo.write('lon,lat,x,y%s'%ls)
82     for (lat,lon),(x0,y0) in zip(data,xy):
83         #print(lat,lon,x0,y0)
84         fo.write('%f,%f,%f,%f%s'%(lat,lon,x0,y0,ls))
85     fo.close()
86     fo=open('rain_time_%s.txt'%fn_ps,'w')
87     fo.write('%d%s'%(len(xy.T[0]),ls))
88     fo.write('x,'+fs.join(['%f'%f for f in xy.T[0]])+ls)
89     fo.write('y,'+fs.join(['%f'%f for f in xy.T[1]])+ls)
90     with zipfile.ZipFile(fn) as myzip:
91         for i0,fn in enumerate(myzip.namelist()[1:]):
92             year,month,day,hour, minu=[int(fn[i1:i2]) for i1,i2 in
[[0,4],[4,6],[6,8],[9,11],[11,13]]]
93             date=datetime.datetime(year,month,day,hour,minu)
94             if i0==0: date0=date
95             #print(fn,date)
96             with myzip.open(fn) as myfile:
97                 zipcont = io.StringIO(myfile.read().decode())
98                 output = np.loadtxt(zipcont,dtype=np.float,delimiter=fs)
99                 output=output[:, -1]
100                 if date == datetime.datetime(2017,7,5,15,10):
101                     plt.pcolor(X,Y,np.where(output < 0.,np.nan,output),cmap=plt
.get_cmap('jet'))

```

```

102         plt.colorbar(label='Rain (mm)')
103         plt.contour(X,Y,np.where(output<0.,np.nan,output),levels
=[50,100],colors=['white'])
104         plt.axes().set_aspect(dx/dy)
105         #plt.show()
106         plt.savefig('{0:%Y/%m/%d_%H%M}.png'.format(date),dpi=300)
107         R2=output[ ij ]
108         if min(R2)<0.:
109             print('{ }<math>\longrightarrow</math>'.format(min(R2)),end='')
110             R2=np.where(R2<0.,0.,R2)
111             print('{ }<math>\longrightarrow</math>'.format(min(R2)), '{0:%Y/%m/%d_%H%M}'.format
(date))
112         R2=R2*dt_min/60. # mm/h -> mm/dt
113         doy=(date-datetime.datetime(date.year,1,1)).days+hour/24.
114         e=eva(doy)
115         nl=fs.join(['%f'%f for f in R2])
116         nl="{0:%Y/%m/%d_%H%M},".format(date)+nl+',%f'%e
117         #print(nl)
118         fo.write(nl+ls)
119         if (datetime.datetime.now()-datetime.datetime(date0)).total_seconds()<dt_min:
120             print('Leap!',date0,'<math>\longrightarrow</math>',date)
121         date0=date
122     fo.close()

```

---

## 4 04\_EXE\_SIMU

### 4.1 01\_RR\_DR

RR\_ver.1.0.f90

```
1 ! Program: RR_ver.1.0.f90
2 ! Copyright (2021) by Public Works Research Institute (P.W.R.I.)
3 ! License: CC-BY-SA
4 !
5 !   1   2   3   end end+1
6 ! FX  q   q   q   q   q
7 !   →  →  →  →  →
8 ! CV | z | z | z |...| z |
9 !   | 1 | 2 | 3 |...|end|
10 !
11 implicit none
12 real(8), parameter :: PI=acos(-1.0d0), deg2rad=PI/180.0d0, rad2deg=180.0d0/PI, &
13     g=9.8d0, dtlim=0.0001d0, cfl=0.01d0
14 real(8), allocatable :: z(:, :), n(:, :), zs(:, :), d(:, :), lamda(:, :), pwc(:, :), &
15     kk(:, :), finf1(:, :), finf2(:, :), d_vash(:, :), facc(:, :))
16 real(8), allocatable :: h(:, :), ho(:, :), hs(:, :), hso(:, :), pw(:, :), pwo(:, :), &
17     eva1(:, :), eva2(:, :), ha(:, :), htmp(:, :), dqd(:, :), dqsd(:, :), &
18     hmax(:, :), hsmax(:, :), grdmax(:, :), grdave(:, :))
19 real(8), allocatable :: qx(:, :), qy(:, :), qxo(:, :), qyo(:, :), &
20     qsx(:, :), qsy(:, :), qsxo(:, :), qsyo(:, :), &
21     qx_acc(:, :), qy_acc(:, :), qsx_acc(:, :), qsy_acc(:, :), &
22     umax(:, :), vmax(:, :), h_umax(:, :), h_vmax(:, :))
23 integer, allocatable :: i_2d(:, :), j_2d(:, :), iriver(:, :), ibasin(:, :), &
24     iacc(:, :), idrain(:, :))
25 logical, allocatable :: m_cv(:, :))
26 integer :: n_ij_cv, n_ij_u, n_ij_v, n_ij_u_we, n_ij_v_sn
27 integer, allocatable :: ij_cv(:, :), ij_u(:, :), ij_v(:, :), ij_u_we(:, :), ij_v_sn(:, :))
28 real(8), allocatable :: rseries(:, :, :), r(:, :), rmap(:, :, :), x_rpnt(:, :), y_rpnt(:, :), &
29     r_rpnt(:, :), d_idw(:, :), eva_tmp(:, :))
30 integer :: n_rpnt, minl(1)
31 character(len=100) :: rname
32 real(8) :: dx, dy, dt, rdt, output_f_time, output_d_time, xllcorner, yllcorner, cellsize, &
33     vad0, d_uni, rn_uni, fs1, fs2, lam_uni, k_uni, pwc_uni, hs0, pw0, hthr
34 real(8) :: time, time0, time1, time2, end_time, dt0, dtdiv, dtmin, dtmin_h2d, flxd, &
35     umax_disp, vmax_disp, h_umax_disp, h_vmax_disp, &
36     r_total, qs_total, q_total, eva1_total, eva2_total, out_total, out_total_ini,
37     finf2_total, &
38     x, y, dist, deg, basin_area, rain_tmp, tmp1
39 integer :: iend, jend, itsp_max, itsp_f_out, itsp_d_out, itsp_rain, rnum
40 integer :: i, j, k, itsp, itr, itr2, maxloc_2d(2), i_umax, j_umax, i_vmax, j_vmax
41 character(len=100) :: fname, cfmt, ctmp, fotime, out_dir
42 character(len=18), allocatable :: datetime(:)
43 ! → 1d
44 real(8), allocatable :: zini_1d(:), z_1d(:), zs_1d(:), b_1d(:), d_1d(:), n_1d(:), &
45     h_1d(:), ho_1d(:), q_in(:), q_out(:), dx_scal(:)
46 real(8), allocatable :: u_1d(:), uo_1d(:), q_1d(:), qo_1d(:), grad_z_1d(:), grad_w_1d(:), dx_vect
47     (:)
48 integer :: ni_inlet, ni_outlet
```



```

47 integer, allocatable :: i_inlet(:), i_outlet(:)
48 !real(8), allocatable :: qin0(:)
49 integer :: ni_scal, ni_vect
50 integer, allocatable :: i_node(:), i_to(:), i_from(:, :), i_scal(:), i_vect(:)
51 integer, allocatable :: ij_2d1d(:, :), iacc_1d(:)
52 logical, allocatable :: mask_1d(:)
53 real(8) :: rn_river, wid_m, wid_p, wid_min, dep_m, dep_p, dep_min, hlim_1d
54 real(8) :: umax_1d, dtmin_h1d, h1, h2, ba, q_1d_total
55 real(8), allocatable :: x_2d_1d(:), y_2d_1d(:), vsl_out(:)
56 integer :: iend_1d, iflg_1d
57 ! <— 1d
58 !
59 open(1, file='RR_input.txt')
60 read(1,*)
61 read(1,*) iend
62 read(1,*) jend
63 read(1,*) dx
64 read(1,*) dy
65 !
66 allocate(z(iend, jend), n(iend, jend), zs(iend, jend), d(iend, jend), &
67         lamda(iend, jend), pwc(iend, jend), kk(iend, jend), finf1(iend, jend), &
68         finf2(iend, jend), d_vash(iend, jend), facc(iend, jend))
69 allocate(h(iend, jend), ho(iend, jend), hs(iend, jend), hso(iend, jend), &
70         pw(iend, jend), pwo(iend, jend), eva1(iend, jend), eva2(iend, jend), &
71         ha(iend, jend), htmp(iend, jend), dqd(iend, jend), dqsd(iend, jend), &
72         hmax(iend, jend), hsmax(iend, jend), grdmax(iend, jend), grdave(iend, jend))
73 allocate(qx(iend+1, jend+1), qy(iend+1, jend+1), qxo(iend+1, jend+1), qyo(iend+1, jend+1), &
74         qsx(iend+1, jend+1), qsy(iend+1, jend+1), qsxo(iend+1, jend+1), qsyo(iend+1, jend+1), &
75         qx_acc(iend+1, jend+1), qy_acc(iend+1, jend+1), qsx_acc(iend+1, jend+1), qsy_acc(iend+1,
76         jend+1), &
77         umax(iend+1, jend+1), vmax(iend+1, jend+1), h_umax(iend+1, jend+1), h_vmax(iend+1, jend
78         +1))
79 allocate(i_2d(iend+1, jend+1), j_2d(iend+1, jend+1), iriver(iend, jend), &
80         ibasin(iend, jend), iacc(iend, jend), idrain(iend, jend), m_cv(iend, jend))
81 !
82 read(1,*)
83 read(1,*) dt
84 read(1,*) output_f_time
85 read(1,*) output_d_time
86 read(1,*)
87 read(1,*) rname
88 read(1,*) rnum
89 read(1,*) rdt
90 !
91 allocate(rseries(iend, jend, 0:rnum))
92 allocate(eva_tmp(0:rnum), datetime(0:rnum))
93 !
94 read(1,*)
95 read(1,*) fname
96 write(*,*) 'elevation', trim(adjustl(fname))
97 call r_fasc(10, fname, z, iend, jend, xllcorner, yllcorner, cellsize)
98 !
99 read(1,*) fname
100 write(*,*) 'targetArea', trim(adjustl(fname))
101 call r_iasc(10, fname, ibasin, iend, jend, xllcorner, yllcorner, cellsize)

```

```

100 basin_area=float(count(ibasin(:,:)/=0))*dx*dx
101 write(*,'(a,f15.3)')'_area_(m2)',basin_area
102 !
103 read(1,*)fname
104 write(*,*)'flowAcc',trim(adjustl(fname))
105 call r_fasc(10,fname,facc,iend,jend,xllcorner,yllcorner,cellsize)
106
107 read(1,*)fname
108 write(*,*)'flowDir',trim(adjustl(fname))
109 call r_iasc(10,fname,idrain,iend,jend,xllcorner,yllcorner,cellsize)
110 !
111 read(1,*)fname
112 write(*,*)'volcanicAsh',trim(adjustl(fname))
113 call r_fasc(10,fname,d_vash,iend,jend,xllcorner,yllcorner,cellsize)
114 read(1,*)vad0 !depth of ash(m) when fs=0
115 !
116 read(1,*)
117 read(1,*)i,d_uni,fname !soil depth(m)
118 if(i/=0)then
119   write(*,*)'d_#####<---_',trim(adjustl(fname))
120   call r_fasc(10,fname,d,iend,jend,xllcorner,yllcorner,cellsize)
121 else
122   d(:,:)=d_uni
123 end if
124 fname=trim(adjustl('output_misc/depth.asc'))
125 cfmt='*(f10.3) '
126 call w_fasc(20,fname,cfmt,d,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)
127 !
128 read(1,*)i,rn_uni,fname !Maning's roughness
129 if(i/=0)then
130   write(*,*)'r_#####<---_',trim(adjustl(fname))
131   call r_fasc(10,fname,n,iend,jend,xllcorner,yllcorner,cellsize)
132 else
133   n(:,:)=rn_uni
134 end if
135 fname=trim(adjustl('output_misc/roughness.asc'))
136 cfmt='*(f10.3) '
137 call w_fasc(20,fname,cfmt,n,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)
138 !
139 read(1,*)i,fs1,fname !surface infiltration(mm/h)
140 if(i/=0)then
141   write(*,*)'fs1(cm/s)_<---_',trim(adjustl(fname))
142   call r_fasc(10,fname,finf1,iend,jend,xllcorner,yllcorner,cellsize)
143   finf1(:,:)=finf1(:,:)/3600.0d0/1000.0d0 !change unit (mm/h) -> (m/s)
144 else
145   finf1(:,:)=fs1/3600.0d0/1000.0d0 !change unit (mm/h) -> (m/s)
146 end if
147 !
148 read(1,*)i,fs2,fname !infiltration to lower layer(mm/h)
149 if(i/=0)then
150   write(*,*)'fs2(cm/s)_<---_',trim(adjustl(fname))
151   call r_fasc(10,fname,finf2,iend,jend,xllcorner,yllcorner,cellsize)
152   finf2(:,:)=finf2(:,:)/3600.0d0/1000.0d0 !change unit (mm/h) -> (m/s)
153 else
154   finf2(:,:)=fs2/3600.0d0/1000.0d0 !change unit (mm/h) -> (m/s)

```

```

155 end if
156 !
157 finf1(:,:)=(-d_vash(:,:)/vad0+1.d0)*finf1(:,:)
158 where( finf1(:,:)<0.d0) finf1(:,:)=0.d0
159 where( ibasin(:,:)/=0) finf2(:,:)=0.d0 !fs2
160 fname=trim(adjustl('output_misc/fs1_mm_h.asc'))
161 cfmt='*(f10.3)'  

162 call w_fasc(20,fname,cfmt,finf1*3.6d+6,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)
163 fname=trim(adjustl('output_misc/fs2_mm_h.asc'))
164 cfmt='*(f10.3)'  

165 call w_fasc(20,fname,cfmt,finf2*3.6d+6,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)
166 !
167 read(1,*)i,lam_uni,fname !porosity
168 if(i/=0)then
169   write(*,*)'lamda_UUUU<---U',trim(adjustl(fname))
170   call r_fasc(10,fname,lamda,iend,jend,xllcorner,yllcorner,cellsize)
171 else
172   lamda(:,:)=lam_uni
173 end if
174 fname=trim(adjustl('output_misc/lamda.asc'))
175 cfmt='*(f10.3)'  

176 call w_fasc(20,fname,cfmt,lamda,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)
177 !
178 read(1,*)i,k_uni,fname !saturated hydraulic conductivity(cm/s)
179 if(i/=0)then
180   write(*,*)'k(cm/s)UUU<---U',trim(adjustl(fname))
181   call r_fasc(10,fname,kk,iend,jend,xllcorner,yllcorner,cellsize)
182 else
183   kk(:,:)=k_uni
184 end if
185 kk=kk/100.d0
186 fname=trim(adjustl('output_misc/k_cm_s.asc'))
187 cfmt='*(e10.3)'  

188 call w_fasc(20,fname,cfmt,kk*1.d+2,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)
189 !
190 read(1,*)i,pwc_uni,fname !clitic vwc for subsurface flow initiation
191 if(i/=0)then
192   write(*,*)'pwcUUUUUU<---U',trim(adjustl(fname))
193   call r_fasc(10,fname,pwc,iend,jend,xllcorner,yllcorner,cellsize)
194 else
195   pwc(:,:)=pwc_uni
196 end if
197 fname=trim(adjustl('output_misc/pwc.asc'))
198 cfmt='*(f10.3)'  

199 call w_fasc(20,fname,cfmt,pwc,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)
200 !
201 read(1,*)i,pw0,fname !initial soil vwc
202 if(i/=0)then
203   write(*,*)'pw_iniUUUU<---U',trim(adjustl(fname))
204   call r_fasc(10,fname,pw,iend,jend,xllcorner,yllcorner,cellsize)
205 else
206   pw(:,:)=pw0
207 end if
208 where( ibasin(:,:)==0)pw(:,:)=0.d0
209 fname=trim(adjustl('output_misc/pw_ini.asc'))

```

```

210 cfmt='*(f10.3)'  

211 call w_fasc(20,fname,cfmt,pw,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)  

212 !  

213 read(1,*)i,hs0,fname !initial hs(m)  

214 if(i/=0)then  

215   write(*,*)'hs_ini_####<---<',trim(adjustl(fname))  

216   call r_fasc(10,fname,hs,iend,jend,xllcorner,yllcorner,cellsize)  

217   else  

218     hs=hs0  

219   end if  

220 where(ibasin(:,:)==0)hs(:,:)=0.d0  

221 fname=trim(adjustl('output_misc/hs_ini.asc'))  

222 cfmt='*(f10.3)'  

223 call w_fasc(20,fname,cfmt,hs,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)  

224 !  

225 zs(:,:)=z(:,:)-d(:,:)  

226 call cal_grad(iend,jend,dx,zs,gradave)  

227 fname=trim(adjustl('output_misc/gradave.asc'))  

228 cfmt='*(f10.3)'  

229 call w_fasc(20,fname,cfmt,gradave*rad2deg,iend,jend,xllcorner,yllcorner,dx,-9999.0d0)  

230 !  

231 read(1,*)  

232 read(1,*)hthr !threshold of water depth to move(m)  

233 !  

234 ! —> stream  

235 read(1,*)  

236 read(1,*)iflg_1d  

237 read(1,*)rn_river  

238 read(1,*)wid_m  

239 read(1,*)wid_p  

240 read(1,*)wid_min  

241 read(1,*)dep_m  

242 read(1,*)dep_p  

243 read(1,*)dep_min  

244 ! <— stream  

245 close(1)  

246 write(*,*)'ctl_read_end'  

247 !  

248 end_time=dble(rnum-1)*rdt  

249 itsp_max=nint(end_time/dt)  

250 itsp_f_out=nint(output_f_time/dt)  

251 itsp_d_out=nint(output_d_time/dt)  

252 itsp_rain=nint(rdt/dt)  

253 !  

254 out_dir='output_RR/'  

255 write(*,*)trim(adjustl(out_dir))  

256 write(*,*)'  

257 ! —> control points etc.  

258 do i=1,iend+1  

259   do j=1,jend+1  

260     i_2d(i,j)=i  

261     j_2d(i,j)=j  

262   end do  

263 end do  

264 !

```

```

265 m_cv(:,:)=(ibasin(:,:)/=0)
266 !
267 open(1, file='input/controlVolume_CenterPoint_inRR.txt')
268 read(1,*) n_ij_cv
269 allocate( ij_cv( n_ij_cv ,2))
270 read(1,*)
271 do k=1, n_ij_cv
272   read(1,*) x, y, ij_cv(k,1:2)
273 end do
274 close(1)
275 open(1, file='input/fluxPoint_x_inRR.txt')
276 read(1,*) n_ij_u
277 allocate( ij_u( n_ij_u ,2))
278 read(1,*)
279 do k=1, n_ij_u
280   read(1,*) x, y, ij_u(k,1:2)
281 end do
282 close(1)
283 open(1, file='input/fluxPoint_y_inRR.txt')
284 read(1,*) n_ij_v
285 allocate( ij_v( n_ij_v ,2))
286 read(1,*)
287 do k=1, n_ij_v
288   read(1,*) x, y, ij_v(k,1:2)
289 end do
290 close(1)
291 open(1, file='input/fluxPoint_xBoundary_inRR.txt')
292 read(1,*) n_ij_u_we
293 allocate( ij_u_we( n_ij_u_we ,8))
294 read(1,*)
295 do k=1, n_ij_u_we
296   read(1,*) x, y, ij_u_we(k,:)
297 end do
298 close(1)
299 open(1, file='input/fluxPoint_yBoundary_inRR.txt')
300 read(1,*) n_ij_v_sn
301 allocate( ij_v_sn( n_ij_v_sn ,8))
302 read(1,*)
303 do k=1, n_ij_v_sn
304   read(1,*) x, y, ij_v_sn(k,:)
305 end do
306 close(1)
307 ! <— control points etc.
308 !
309 ! —> rain and eva
310 open(11, file=name)
311 read(11,*) n_rpnt
312 allocate( r(iend, jend), rmap(iend, jend, 0:n_rpnt), x_rpnt(n_rpnt), y_rpnt(n_rpnt), &
313          r_rpnt(n_rpnt), d_idw(n_rpnt))
314 read(11,*) ctmp, x_rpnt(1:n_rpnt)
315 read(11,*) ctmp, y_rpnt(1:n_rpnt)
316 rseries(:, :, 0:rnum)=0.0d0 ! from 0
317 rmap(:, :, :)=0
318 do i=1, iend
319   do j=1, jend

```

```

320   if (ibasin(i,j)/=0)then
321     x=xllcorner+dx*(0.5+i-1)
322     y=yllcorner+dx*(0.5+j-1)
323     d_idw(1:n_rpnt)=((x-x_rpnt(1:n_rpnt))**2.d0+(y-y_rpnt(1:n_rpnt))**2.d0)**0.5d0
324     minl=minloc(d_idw)
325     !write(*,*)i,j,minl
326     rmap(i,j,1:n_rpnt)=0.d0
327     rmap(i,j,minl(1))=1.d0 !d_idw(1:n_rpnt)/sum(d_idw(:))
328     rmap(i,j,0)=minl(1)
329   end if
330 end do
331 end do
332 fname='output_misc/rain_map.asc'
333 cfmt='*(f10.3)''
334 call w_fasc(20,fname,cfmt,rmap(:,:0),iend,jend,xllcorner,yllcorner,dx,-9999.d0)
335 !
336 do k=1,n_rpnt
337   write(ctmp,'(i4.4)')k
338   fname='output_misc/rain_map'//trim(ctmp)//'.asc'
339   cfmt='*(f10.3)''
340   !call w_fasc(20,fname,cfmt,rmap(:,:),k,iend,jend,xllcorner,yllcorner,dx,-9999.d0)
341 end do
342 !
343 open(20,file='output_misc/basin_rain.txt')
344 write(20,'(a)')'datetime,rain(mm),maxval'
345 !
346 rseries(:,:)=0.0d0
347 do k=0,rnum-1
348   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
349   read(11,*)datetime(k),r_rpnt(1:n_rpnt),eva_tmp(k)
350   if(k==0 .or. k==rnum)write(*,'(x,a,a,i5)')datetime(k),'_rain_#_',k
351   eva_tmp(k)=max(eva_tmp(k),0.d0)/rdt/1000.0d0
352   do i=1,iend
353     do j=1,jend
354       if (ibasin(i,j)/=0)then
355         !rain_tmp=sum(r_rpnt(1:n_rpnt)*rmap(i,j,1:n_rpnt))
356         rain_tmp=r_rpnt(int(rmap(i,j,0)))
357         !write(*,'(2i4,10f7.3)')i,j,rain_tmp,r_rpnt(:),rmap(i,j,:)
358         rseries(i,j,k)=rain_tmp/rdt/1000.0d0
359       end if
360     end do
361   end do
362   !write(*,*)datetime(k),maxval(rseries(:,:),k),mask=ibasin/=0)*rdt*1000.d0
363   write(20,'(a,2(a,f10.3))')trim(adjustl(datetime(k))),',', '&
364   sum(rseries(:,:),k),mask=ibasin/=0)*dx*dx/basin_area*rdt*1000.0d0,',',maxval(rseries(:,:),
365   k),mask=ibasin/=0)*rdt*1000.d0
366   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
366   !rseries(:,:),k)=rain_tmp/rdt/1000.0d0
367   !where(ibasin(:,:)=0) rseries(:,:),k)=0.0d0
368   !write(fname,'(i10.10)')nint(k*rdt)
369   !fname='output_rain/rain_'//trim(adjustl(fname))//'.asc'
370   !write(*,*)trim(adjustl(fname)), ' mm/h'
371   !cfmt='*(f10.3)''
372   !call w_fasc(20,cfmt,rseries(:,:),k)*3600.*1000.0d0,iend,jend,xllcorner,yllcorner,dx,-9999.0
    d0)

```

```

373 !
374 !open(10,file=fname)
375 !do i=1,6
376 ! read(10,*)
377 !end do
378 !do j=jend,1,-1
379 ! read(10,*)rseries(1:iend,j,k)
380 ! rseries(1:iend,j,k)=rseries(1:iend,j,k)/rdt/1000.0d0
381 !end do
382 !
383 end do
384 close(11)
385 close(20)
386 !
387 fname='output_misc/rain_sum.asc'
388 r=0.d0
389 do k=0,rnum-1
390 r(:,:)=r(:,:)+(rseries(:,:,k))*rdt*1000.
391 end do
392 write(*,'(a,f10.3,a)')'_basin_ave_ rain: ',sum(r(:,:))*dx*dx/basin_area,'_mm'
393 write(*,*)''
394 cfmt='*(f10.3)''
395 call w_fasc(20,fname,cfmt,r(:,:),iend,jend,xllcorner,yllcorner,dx,-9999.0d0)
396 r=0.d0
397 rseries(:,:,rnum)=rseries(:,:,rnum-1)
398 datetime(rnum)=datetime(rnum-1)
399 eva_tmp(rnum)=eva_tmp(rnum-1)
400 ! <— rain and eva
401 !
402 ! —> 1d
403 open(10,file='input/streamConfiguration_inRR.txt')
404 read(10,*)iend_1d
405 i=iend_1d
406 !
407 allocate(zini_1d(i),z_1d(i),zs_1d(i),b_1d(i),d_1d(i),n_1d(i),h_1d(i),ho_1d(i), &
408         q_in(i),q_out(i),iacc_1d(i),dx_scal(i))
409 allocate(u_1d(i),uo_1d(i),q_1d(0:i),qo_1d(0:i),grad_z_1d(i),grad_w_1d(i),dx_vect(i))
410 !allocate(qin0(0:qnum),cin0(0:qnum))
411 allocate(i_node(i),i_to(i),i_from(i,3),mask_1d(i))
412 allocate(ij_2d1d(i,2),x_2d_1d(i),y_2d_1d(i),vsl_out(i))
413 !
414 read(10,*)
415 do i=1,iend_1d
416 read(10,*)i_node(i),i_to(i),i_from(i,1),i_from(i,2),i_from(i,3), &
417         tmp1,z_1d(i),zs_1d(i),dx_scal(i),b_1d(i),n_1d(i),iacc_1d(i), &
418         x_2d_1d(i),y_2d_1d(i),ij_2d1d(i,1),ij_2d1d(i,2)
419 dx_vect(i)=dx_scal(i)
420 tmp1=z(ij_2d1d(i,1),ij_2d1d(i,2))
421 if (iflg_1d==0)then
422 ba=cellsize*cellsize*float(iacc_1d(i))/1000000.d0
423 b_1d(i)=max(wid_min,wid_m*ba**wid_p)
424 d_1d(i)=max(dep_min,dep_m*ba**dep_p)
425 z_1d(i)=tmp1-d_1d(i)
426 n_1d(i)=rn_river
427 end if

```

```

428 end do
429 close(10)
430 !
431 zini_1d(:)=z_1d(:)
432 mask_1d=(i_from(:,1)/=0)!.and. i_to(:)/=0)
433 ni_scal=count(mask_1d)
434 allocate(i_scal(ni_scal))
435 i_scal(1:ni_scal)=pack(i_node(:),mask_1d)
436 open(20,file='output_misc/scal_point.txt')
437 write(20,*)'x(s(i)),y(s(i)),s(i),i'
438 do i=1,ni_scal
439   write(20,'(2(f15.5,:',','),2(i5,:',','))')x_2d_1d(i_scal(i)),y_2d_1d(i_scal(i)),i_scal(i),
      i
440 end do
441 close(20)
442 !
443 mask_1d=i_to(:)/=0
444 ni_vect=count(mask_1d)
445 allocate(i_vect(ni_vect))
446 i_vect(1:ni_vect)=pack(i_node(:),mask_1d)
447 open(20,file='output_misc/vect_point.txt')
448 write(20,*)'x(v(i)),y(v(i)),dist,deg,i,i_to'
449 do i=1,ni_vect
450   x=x_2d_1d(i_vect(i))
451   y=y_2d_1d(i_vect(i))
452   x=(x+x_2d_1d(i_to(i_vect(i))))*0.5d0
453   y=(y+y_2d_1d(i_to(i_vect(i))))*0.5d0
454   h1=x_2d_1d(i_to(i_vect(i)))-x_2d_1d(i_vect(i)) !dx->h1
455   h2=y_2d_1d(i_to(i_vect(i)))-y_2d_1d(i_vect(i)) !dy->h2
456   dist=(h1*h1+h2*h2)**0.5d0
457   h1=h1/dist !dx->h1
458   h2=h2/dist !dy->h2
459   if(0<h1)then
460     deg=atan(h2/h1)*rad2deg
461   else if(h1<0)then
462     deg=atan(h2/h1)*rad2deg
463   else
464     deg=90.
465   end if
466   deg=deg*(-45.d0)+90.d0
467   write(20,'(2(f15.5,:',','),2(f10.3,:',','),2(i5,:',','))')&
468     x,y,dist,deg,i_vect(i),i_to(i_vect(i))
469 end do
470 close(20)
471 !
472 mask_1d=i_from(:,1)==0
473 ni_inlet=count(mask_1d)
474 allocate(i_inlet(ni_inlet))
475 i_inlet(1:ni_inlet)=pack(i_node(:),mask_1d)
476 open(20,file='output_misc/inlet_point.txt')
477 write(20,*)'x(s(i)),y(s(i)),s(i),i'
478 do i=1,ni_inlet
479   write(20,'(2(f15.5,:',','),2(i5,:',','))')x_2d_1d(i_inlet(i)),y_2d_1d(i_inlet(i)),i_inlet(
      i),i
480 end do

```



```

481 close(20)
482 !
483 mask_1d=i_to(:)==0
484 ni_outlet=count(mask_1d)
485 allocate(i_outlet(ni_outlet))
486 i_outlet(1:ni_outlet)=pack(i_node(:),mask_1d)
487 open(20,file='output_misc/outlet_point.txt')
488 write(20,*)'x(s(i)),y(s(i)),s(i),i'
489 do i=1,ni_outlet
490   write(20,'(2(f15.5,:',','),2(i5,:',','))')x_2d_1d(i_outlet(i)),y_2d_1d(i_outlet(i)),
      i_outlet(i),i
491 end do
492 close(20)
493 !
494 u_1d(:)=0.d0
495 uo_1d(:)=u_1d(:)
496 h_1d(:)=0.d0
497 ho_1d(:)=h_1d(:)
498 q_1d(:)=0.d0
499 qo_1d(:)=q_1d(:)
500 q_in(:)=0.d0
501 q_out(:)=0.d0
502 hlim_1d=hthr
503 ! ← 1d
504 !
505 ! → initial conditions
506 h(:,:)=0.d0
507 ho(:,:)=h(:,:)
508 hso(:,:)=hs(:,:)
509 hmax(:,:)=0.d0
510 hsmax(:,:)=0.d0
511 pwo(:,:)=pw(:,:)
512 eva1(:,:)=0.d0
513 eva2(:,:)=0.d0
514 !
515 qx(:,:)=0.0d0
516 qy(:,:)=0.0d0
517 qxo(:,:)=qx(:,:)
518 qyo(:,:)=qy(:,:)
519 qsx(:,:)=0.0d0
520 qsy(:,:)=0.0d0
521 qsxo(:,:)=qsx(:,:)
522 qsy(:,:)=qsy(:,:)
523 qx_acc(1:iend+1,1:jend+1)=0.0d0
524 qy_acc(1:iend+1,1:jend+1)=0.0d0
525 qsx_acc(1:iend+1,1:jend+1)=0.0d0
526 qsy_acc(1:iend+1,1:jend+1)=0.0d0
527 !
528 r_total=0.0d0
529 q_total=0.0d0
530 qs_total=0.d0
531 eva1_total=0.d0
532 eva2_total=0.d0
533 finf2_total=0.d0
534 q_1d_total=0.d0

```

```

535 out_total_ini=finf2_total+qs_total+q_total+eva1_total+eva2_total+(sum(h)+sum(hs*(lamda-pwc
    ))+sum(pw*d))*dx*dx
536 out_total=0.d0
537 vsl_out(:)=0.d0
538 umax_disp=0.d0
539 vmax_disp=0.d0
540 h_umax_disp=0.d0
541 h_vmax_disp=0.d0
542 maxloc_2d=0
543 i_umax=0
544 j_umax=0
545 i_vmax=0
546 j_vmax=0
547 dtmin=dt
548 dqd=0.d0
549 dqsd=0.d0
550 ! <— initial conditions
551 !
552 dt0=dt
553 dtmin=dt
554 time=0.0d0
555 time0=0.d0
556 open(200, file='RR_summary.txt')
557 write(200, '(10a21,4a11,a10)')&
558 'time,', 'rain,', 'out_t,', 'out_qs,', 'out_q,', 'out_eval1,', 'out_eval2,', &
559 'sum_hs,', 'sum_h,', 'sum_pw,', 'slope,', 'debris,', 'fs2,', 'out_q1d', 'out_h1d'
560 open(201, file='flowVolume_RRtoDR.txt')
561 write(201, '(a10,"",a10,(i12,:","))')',',', 'i_2d', ij_2d1d(:,1)
562 write(201, '(a10,"",a10,(i12,:","))')',',', 'j_2d', ij_2d1d(:,2)
563 write(200, '(a,i10,9(a,E20.10e3),2(a,i10),6(a,E20.10e3))')&
564 'time(s)=', nint(time),',',',', r_total,',',',', out_total, &
565 ',',',', qs_total,',',',', q_total,',',',', eva1_total,',',',', eva2_total,',',',', &
566 sum(hs*(lamda-pwc))*dx*dx,',',',', sum(h)*dx*dx,',',',', sum(pw*d)*dx*dx,',',',', &
567 0,',',',', 0,',',',', finf2_total,',',',', 0.d0,',',',', 0.d0
568 write(201, '(a10,"",i10,(e12.3,:","))')',',', 'time(s)=', nint(time), vsl_out(:)
569 !
570 write(*,*) 't=0_--->', datetime(0)
571 write(*,*) ''
572 !
573 do itsp=1,itsp_max
574   time=time0+dt
575   k=itsp/itsp_rain
576   ! —> rain
577   ! r(:, :) = rseries(:, :, k) + (rseries(:, :, k+1) - rseries(:, :, k)) * (itsp - k * itsp_rain) / dble(itsp_rain)
578   r(:, :) = rseries(:, :, k) ! r=0,1,2,...
579   where(ibasin(:, :) == 0) r(:, :) = 0.d0
580   ! <— rain
581   ! —> eva
582   eva1(:, :) = eva_tmp(k) + (eva_tmp(k+1) - eva_tmp(k)) * (itsp - k * itsp_rain) / dble(itsp_rain)
583   where(ibasin(:, :) == 0) eva1(:, :) = 0.d0
584   eva2(:, :) = eva1(:, :)
585   ! <— eva
586   time1=time0
587   itr=1

```

```

588 dt=dt0
589 umax(:,:)=0.d0
590 vmax(:,:)=0.d0
591 h_umax(:,:)=0.d0
592 h_vmax(:,:)=0.d0
593 flxd=0.d0
594 call cal_q(n_ij_u, ij_u, iend, jend, z, h+r*dt, n, -1, 0, hthr, dx, qx, umax, h_umax, dt, 0.d0)
595 call cal_q(n_ij_v, ij_v, iend, jend, z, h+r*dt, n, 0, -1, hthr, dx, qy, vmax, h_vmax, dt, 0.d0)
596 flxd=max(flxd, maxval(umax)*dt, maxval(vmax)*dt)
597 call cal_qs(n_ij_u, ij_u, iend, jend, zs, hs, -1, 0, kk, hthr, dx, qsx, umax, h_umax)
598 call cal_qs(n_ij_v, ij_v, iend, jend, zs, hs, 0, -1, kk, hthr, dx, qsy, vmax, h_vmax)
599 flxd=max(flxd, maxval(umax)*dt, maxval(vmax)*dt)
600 dtmin_h2d=dt
601 call cal_dq(n_ij_cv, ij_cv, iend, jend, dx, h, qx, qy, dqd, dtmin_h2d) ! h as dq
602 if(dt < dtmin_h2d)then
603     write(*,*)'dt<dtmin_h2d', dt, dtmin_h2d
604     dt=dtmin_h2d
605     read(*,*)
606 end if
607 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
608 if ((dx*cfl) < flxd .or. dtmin_h2d < dt)then
609     itr=max(1, ceiling(flxd/(dx*cfl)), nint(dt0/dtmin_h2d))
610     if(itr<=0)then
611         write(*,*)'itr', itr, flxd, dx*cfl, nint(dt0/dtlim)
612         read(*,*)
613     end if
614     dt=dt0/dble(itr)
615     write(*, '(a,f15.7,a,i10)', advance='yes')'2d_dt=', dt, ',_itr=', itr
616     if(dt<dtmin)dtmin=dt
617 end if
618 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
619 do while (itr>0)
620     ! —> cal 2D
621     call cal_dq(n_ij_cv, ij_cv, iend, jend, dx, h, qx, qy, dqd, 0.d0) ! h as dq
622     call cal_dq(n_ij_cv, ij_cv, iend, jend, dx, hs, qsx, qsy, dqsd, 0.d0) ! hs as dqsd
623     where(m_cv(:,:)) htmp(:,:)=min(ho(:,:)+(r(:,:)+dqd(:,:))*dt, finf1(:,:)*dt)
624     where(m_cv(:,:)) ha(:,:)=htmp(:,:)+(lamda(:,:)-pwc(:,:))*hs(:,:)+d(:,:)*pw(:,:)+&
625     !dt*(dqsd(:,:)-eva1(:,:)-eva2(:,:)) ! dt!
626     dt*(dqsd(:,:)) ! dt!
627     !where(m_cv(:,:)) finf2(:,:)=fs2
628     where(m_cv(:,:)) finf2(:,:)=fs2*hs(:,:)
629     call cal_divha(n_ij_cv, ij_cv, iend, jend, lamda, pwc, ha, h, hs, pw, d, finf2, dt, qx, qy, qsx, qsy)
630     where(m_cv(:,:)) htmp(:,:)=max(ho(:,:)+(r(:,:)+dqd(:,:))*dt-finf1(:,:)*dt, 0.d0)
631     h(:,:)=h(:,:)+htmp(:,:)
632     ! —> cal eva
633     call cal_eva(n_ij_cv, ij_cv, iend, jend, r, hs, pw, lamda, pwc, d, dt, eva1, eva2)
634     ! <— cal eva
635     !
636     ! —> cal flow from 2d to 1d
637     call cal_2d1d(iend, jend, ij_2d1d, iend_1d, dx, dt, z, h, z_1d, h_1d, b_1d, dx_scal, vs1_out)
638     ! <— cal flow from 2d to 1d
639     !
640     ! —> cal surface flux
641     call cal_q(n_ij_u, ij_u, iend, jend, z, h, n, -1, 0, hthr, dx, qx, umax, h_umax, dt, cfl)
642     call cal_q(n_ij_v, ij_v, iend, jend, z, h, n, 0, -1, hthr, dx, qy, vmax, h_vmax, dt, cfl)

```

```

643  ! ← cal surface flux
644  !
645  ! → cal subsurface flux
646  call cal_qs(n_ij_u , ij_u , iend , jend , zs , hs , -1 , 0 , kk , hthr , dx , qsx , umax , h_umax)
647  call cal_qs(n_ij_v , ij_v , iend , jend , zs , hs , 0 , -1 , kk , hthr , dx , qsy , vmax , h_vmax)
648  ! ← cal subsurface flux
649  !
650  qx_acc( 1:iend+1,1:jend+1)=qx_acc( 1:iend+1,1:jend+1)+qx( 1:iend+1,1:jend+1)*dx*dt
651  qy_acc( 1:iend+1,1:jend+1)=qy_acc( 1:iend+1,1:jend+1)+qy( 1:iend+1,1:jend+1)*dx*dt
652  qsx_acc(1:iend+1,1:jend+1)=qsx_acc(1:iend+1,1:jend+1)+qsx(1:iend+1,1:jend+1)*dx*dt
653  qsy_acc(1:iend+1,1:jend+1)=qsy_acc(1:iend+1,1:jend+1)+qsy(1:iend+1,1:jend+1)*dx*dt
654  ho(:, :)=h(:, :)
655  hso(:, :)=hs(:, :)
656  pwo(:, :)=pw(:, :)
657  qxo(:, :)=qx(:, :)
658  qyo(:, :)=qy(:, :)
659  qsxo(:, :)=qsx(:, :)
660  qsyo(:, :)=qsy(:, :)
661  where(hmax(:, :)<h(:, :))hmax(:, :)=h(:, :)
662  where(hsmax(:, :)<hs(:, :))hsmax(:, :)=hs(:, :)
663  eva1_total=eva1_total+sum(eva1)*dx*dx*dt
664  eva2_total=eva2_total+sum(eva2)*dx*dx*dt
665  finf2_total=finf2_total+sum((pack(finf2(:, :), mask=m_cv(:, :))) * dt) * dx * dx
666  do k=1, n_ij_u_we
667     qs_total=qs_total+abs(-qsx(ij_u_we(k,1), ij_u_we(k,2))) * dx * dt
668     q_total =q_total +abs(- qx(ij_u_we(k,1), ij_u_we(k,2))) * dx * dt
669  end do
670  do k=1, n_ij_v_sn
671     qs_total=qs_total+abs(qsy(ij_v_sn(k,1), ij_v_sn(k,2))) * dx * dt
672     q_total =q_total +abs( qy(ij_v_sn(k,1), ij_v_sn(k,2))) * dx * dt
673  end do
674  r_total=r_total+sum(r(:, :)) * dx * dx * dt
675  out_total=-out_total_ini+finf2_total+qs_total+q_total+eva1_total+eva2_total+(sum(h)+sum(
hs*(lamda-pwc))+sum(pw*d)) * dx * dx
676  if(umax_disp<maxval(umax))then
677     umax_disp=maxval(umax)
678     maxloc_2d=maxloc(umax)
679     i_umax=maxloc_2d(1)
680     j_umax=maxloc_2d(2)
681     h_umax_disp=h_umax(i_umax, j_umax)
682  end if
683  if(vmax_disp<maxval(vmax))then
684     vmax_disp=maxval(vmax)
685     maxloc_2d=maxloc(vmax)
686     i_vmax=maxloc_2d(1)
687     j_vmax=maxloc_2d(2)
688     h_vmax_disp=h_vmax(i_vmax, j_vmax)
689  end if
690  ! ← cal 2D
691  !
692  ! → cal 1d
693  time2=time1
694  itr2=1
695  dtdiv=dt
696  umax_1d=0.d0

```

```

697  call cal_1d_uhb(ni_vect, i_vect, iend_1d, i_to, z_1d, h_1d, b_1d, n_1d, dtdiv, 0.d0, dx_vect, u_1d,
u_o_1d, hlim_1d, umax_1d, q_1d)
698  u_1d(i_outlet(:))=u_1d(i_from(i_outlet(:),1))
699  flxd=umax_1d*dtdiv
700  q_1d(i_outlet(:))=b_1d(i_outlet(:))*h_1d(i_outlet(:))*u_1d(i_outlet(:))
701  q_in(i_scal(:))=q_1d(i_from(i_scal(:),1))+q_1d(i_from(i_scal(:),2))+q_1d(i_from(i_scal
(:),3))
702  q_in(i_inlet(1:ni_inlet))=0.d0 !qin
703  q_out(:)=q_1d(i_node(:))
704  dtmin_h1d=dtdiv
705  ho_1d=h_1d
706  call cal_1d_h(iend_1d, i_node, iend_1d, dtdiv, dx_scal, b_1d, h_1d, iend_1d, q_in, q_out,
dtmin_h1d)
707  h_1d=ho_1d
708  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
709  if ((dx*cfl) < flxd .or. dtmin_h1d < dt )then
710     itr2=max(nint(dt/dtmin_h1d), ceiling(flxd/(dx*cfl)))
711     if(itr2<=0)then
712        write(*,*)'itr2', itr2, flxd, dx*cfl, nint(dt0/dtlim)
713        read(*,*)
714     end if
715     dtdiv=dt/dble(itr2)
716     if(dtdiv<dtmin)dtmin=dtdiv
717  end if
718  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
719  do while (itr2>0)
720     dtmin_h1d=0.d0
721     call cal_1d_h(iend_1d, i_node, iend_1d, dtdiv, dx_scal, b_1d, h_1d, iend_1d, q_in, q_out,
dtmin_h1d)
722     call cal_1d_uhb(ni_vect, i_vect, iend_1d, i_to, z_1d, h_1d, b_1d, n_1d, dtdiv, cfl, dx_vect, u_1d
, u_o_1d, hlim_1d, umax_1d, q_1d)
723     u_1d(i_outlet(:))=u_1d(i_from(i_outlet(:),1))
724     q_1d(i_outlet(:))=b_1d(i_outlet(:))*h_1d(i_outlet(:))*u_1d(i_outlet(:))
725     q_in(i_scal(:))=qo_1d(i_from(i_scal(:),1))+qo_1d(i_from(i_scal(:),2))+qo_1d(i_from(
i_scal(:),3))
726     q_in(i_inlet(1:ni_inlet))=0.d0 !qin
727     q_out(:)=qo_1d(i_node(:))
728     uo_1d(:)=u_1d(:)
729     ho_1d(:)=h_1d(:)
730     qo_1d(:)=q_1d(:)
731     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
732     q_1d_total=q_1d_total+sum(q_out(i_from(i_outlet(:),1)))*dtdiv
733     h_1d(i_outlet(1:ni_outlet))=0.d0
734     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
735     time2=time2+dtdiv
736     itr2=itr2-1
737  end do ! div, do while
738  ! <— cal 1d
739  !
740  time1=time1+dt
741  itr=itr-1
742  end do ! do while
743  time0=time
744  dt=dt0
745  ! <— output

```

```

746 if (mod(itssp, itssp_d_out) == 0) then
747   ! —> disp out
748   write(*,*)
749   write(*, '(i5,a,a,a,i5,a,a)') itssp/itssp_rain, ' ', datetime(itssp/itssp_rain), ' ', &
750   int(mod(itssp, itssp_rain)*dt0), ' (s) ---> ', datetime(itssp/itssp_rain+1)
751   write(*, '(a,f10.4,a,f10.3)') 'time(day) = ', time/86400.d0, ' ', 'time(hour) = ', time/3600.d0
752   write(*, '(a,i10,a,f10.3,a,f6.2,a,2i4)') &
753     'time(s) = ', nint(time), ' ', 'rain(mm/h) = ', maxval(r(:, :))*1000.d0*3600.d0
754   write(*, '(2(a,f6.2,a,2i4,a)') &
755     'qx_max = ', maxval(abs(qx)), ' ', 'i, j = ', maxloc(abs(qx)), ' ', &
756     'qy_max = ', maxval(abs(qy)), ' ', 'i, j = ', maxloc(abs(qy))
757   write(*, '(2(a,f6.2,a,2i4,a)') &
758     'qsx_max = ', maxval(abs(qsx)), ' ', 'i, j = ', maxloc(abs(qsx)), ' ', &
759     'qsy_max = ', maxval(abs(qsy)), ' ', 'i, j = ', maxloc(abs(qsy))
760   write(*, '(2(a,f6.2,a,2i4,a)') &
761     'uh_max = ', maxval(h), ' ', 'i, j = ', maxloc(h), ' ', &
762     'hs_max = ', maxval(hs), ' ', 'i, j = ', maxloc(hs)
763   write(*, '(2(a,f10.3),a,2i5)') 'umax = ', umax_disp, ' ', 'uh@umax = ', h_umax_disp, ' ', 'i, j = ', i_umax,
764     j_umax
765   write(*, '(2(a,f10.3),a,2i5)') 'vmax = ', vmax_disp, ' ', 'uh@vmax = ', h_vmax_disp, ' ', 'i, j = ', i_vmax,
766     j_vmax
767   write(*, '(2(a,f10.5)') 'dt_min = ', dtmin, ' ', 'dt_lim = ', dtlim
768   umax_disp = 0.d0
769   vmax_disp = 0.d0
770   dtmin = dt0
771   write(*, '(4(a,e12.5)') &
772     'rain = ', r_total, ' ', 'pw = ', sum(pw*d)*dx*dx, ' ', 'e1 = ', eval_total, ' ', 'e2 = ', eva2_total
773   write(*, '(4(a,e12.5)') &
774     'hs = ', sum(hs*(lamda-pwc))*dx*dx, ' ', 'uh = ', (sum(h))*dx*dx, ' ', 'qs = ', qs_total, ' ', 'q
775     = ', q_total
776   write(*, '(3(a,e12.5)') 'sum = ', q_1d_total+out_total+sum(h_1d*b_1d*dx_scal), &
777     'h_1d = ', sum(h_1d*b_1d*dx_scal), ' ', 'q_1d = ', q_1d_total
778   !do i=1,ni_outlet
779     !write(*, '(2(a,e12.5)') 'q_1d = ', q_1d_total, ' ', 'h_1d = ', sum(h_1d*b_1d*dx_scal)
780   !end do
781   write(*,*)
782   ! <— disp out
783   write(200, '(a,i10,9(a,E20.10e3),2(a,i10),6(a,E20.10e3)') &
784     'time(s) = ', nint(time), ' ', 'r_total = ', &
785     out_total, ' ', 'qs_total = ', ' ', 'q_total = ', ' ', &
786     eval_total, ' ', 'eva2_total = ', ' ', sum(hs*(lamda-pwc))*dx*dx, ' ', 'sum(h)*dx*dx = ', ' ', &
787     sum(pw*d)*dx*dx, ' ', '0 = ', ' ', '0 = ', ' ', &
788     finf2_total, ' ', 'q_1d_total = ', ' ', sum(h_1d*b_1d*dx_scal)
789   write(201, '(a10,"",i10,*(e12.3,:","))') 'time(s) = ', nint(time), vsl_out(:)
790 end if
791 !
792 if (mod(itssp, itssp_f_out) == 0 .or. itssp==itssp_max) then
793   write(fotime, '(i10.10)') nint(time)
794   ! —> 1d output
795   fname = trim(adjustl(out_dir))// 'res_1d_'// trim(adjustl(fotime))// '.txt'
796   open(20, file=fname)
797   do i=1,iend_1d
798     write(20, '(2i5,i10,3f10.3,f15.3)') ij_2d1d(i,1), ij_2d1d(i,2), i, z_1d(i), h_1d(i), u_1d(i),
799       q_1d(i)
800   end do

```

```

797  close(20)
798  ! <— Id output
799  cfmt='*(f10.3)'  

800  fname=trim(adjustl(out_dir))//'res_2d_hs_'//trim(adjustl(fotime))//'.asc'  

801  call w_fasc(20,fname,cfmt,hs(:,:),iend,jend,xllcorner,yllcorner,dx,0.d0)  

802  !  

803  fname=trim(adjustl(out_dir))//'res_2d_h_'//trim(adjustl(fotime))//'.asc'  

804  call w_fasc(20,fname,cfmt,h(:,:),iend,jend,xllcorner,yllcorner,dx,0.d0)  

805  !  

806  !  fname='output/pw_ '//trim(adjustl(fotime))//'.asc'  

807  !  call w_fasc(20,fname,cfmt,pw(:,:),iend,jend,xllcorner,yllcorner,dx,0.d0)  

808  !  !  

809  fname=trim(adjustl(out_dir))//'res_2d_hmax.asc'  

810  call w_fasc(20,fname,cfmt,hmax(:,:),iend,jend,xllcorner,yllcorner,dx,0.d0)  

811  !  !  

812  fname=trim(adjustl(out_dir))//'res_2d_hsmax.asc'  

813  call w_fasc(20,fname,cfmt,hsmax(:,:),iend,jend,xllcorner,yllcorner,dx,0.d0)  

814  !  !  

815  !  fname='output/qx_ave_ '//trim(adjustl(fotime))//'.asc'  

816  !  call w_fasc(20,fname,cfmt,qx_acc(:,:)/output_f_time,iend+1,jend+1, &  

817  !  xllcorner-dx*0.5,yllcorner,dx,0.d0)  

818  !  !  

819  !  fname='output/qy_ave_ '//trim(adjustl(fotime))//'.asc'  

820  !  call w_fasc(20,fname,cfmt,qy_acc(:,:)/output_f_time,iend+1,jend+1, &  

821  !  xllcorner,yllcorner-dx*0.5,dx,0.d0)  

822  !  !  

823  !  fname='output/qsx_ave_ '//trim(adjustl(fotime))//'.asc'  

824  !  call w_fasc(20,fname,cfmt,qsx_acc(:,:)/output_f_time,iend+1,jend+1, &  

825  !  xllcorner-dx*0.5,yllcorner,dx,0.d0)  

826  !  !  

827  !  fname='output/qsy_ave_ '//trim(adjustl(fotime))//'.asc'  

828  !  call w_fasc(20,fname,cfmt,qsy_acc(:,:)/output_f_time,iend+1,jend+1, &  

829  !  xllcorner,yllcorner-dx*0.5,dx,0.d0)  

830  !  

831  qx_acc(:,:)=0.0d0  

832  qy_acc(:,:)=0.0d0  

833  qsx_acc(:,:)=0.0d0  

834  qsy_acc(:,:)=0.0d0  

835  !  

836  end if  

837  ! <— output  

838  !  

839  end do  

840  !  

841  write(*,*)'---_nomal_end_---'  

842  !  

843  stop  

844  end  

845  !  

846  !  

847  !  

848  subroutine cal_q(n_1d,ij_1d, &  

849  iend,jend,z,h,n,ioffset,joffset,hthr,dx,flux,uvmax,h_uvmax,dt,cfl)  

850  use omp_lib  

851  implicit none

```

```

852 real(8), parameter :: hm=5.0d0/3.0d0
853 integer :: ni,n_1d,ij_1d(1:n_1d,1:2),i,j,iend,jend,ip,jp,ioffset,joffset
854 real(8) :: z(iend,jend),h(iend,jend),wl(iend,jend),n(iend,jend), &
855         flux(iend+1,jend+1),uvmax(iend+1,jend+1),h_uvmax(iend+1,jend+1), &
856         grad,htmp,uv,dt,cfl,hthr,dx,cost,sint
857 !
858 !uvmax=0.d0
859 wl=z+h
860 !$omp parallel do private(ni,i,j,ip,jp,grad,cost,sint,htmp,uv)
861 do ni=1,n_1d
862     i=ij_1d(ni,1)
863     j=ij_1d(ni,2)
864     ip=i+ioffset
865     jp=j+joffset
866     grad=(wl(i,j)-wl(ip,jp))/dx
867     grad=sign(sin(atan(abs(grad))),grad)
868     cost=cos(atan(grad))
869     sint=sin(atan(grad))
870     if(grad>0.0d0)then
871         if(wl(i,j)<z(ip,jp) .or. h(i,j)<hthr)then
872             flux(i,j)=0.0d0
873         else
874             htmp=wl(i,j)-max(z(i,j),z(ip,jp))
875             htmp=htmp*cost
876             flux(i,j)=sign(1.0d0/n(i,j)*abs(sint)**0.5d0*htmp**hm,-grad)
877             uv=abs(flux(i,j))/htmp
878         end if
879     else
880         if(wl(ip,jp)<z(i,j) .or. h(ip,jp)<hthr )then
881             flux(i,j) = 0.0d0
882         else
883             htmp=wl(ip,jp)-max(z(i,j),z(ip,jp))
884             htmp=htmp*cost
885             flux(i,j)=sign(1.0d0/n(ip,jp)*abs(sint)**0.5d0*htmp**hm,-grad)
886             uv=abs(flux(i,j))/htmp
887         end if
888     end if
889     if(dx/dt*cfl < uv .and. 0.d0<cfl)then
890         uv=sign(dx/dt*cfl,flux(i,j))
891         flux(i,j)=uv*htmp
892     end if
893     if(uvmax(i,j)<uv)then
894         uvmax(i,j)=uv
895         h_uvmax(i,j)=htmp
896     end if
897 end do
898 !$omp end parallel do
899 !call cal_q_bd(n_1d_bd,ij_1d_bd,iend,jend,z,h,n,dd,flux)
900 !
901 end subroutine cal_q
902 !
903 !
904 !
905 subroutine cal_q_bd(n_1d,ij_1d,iend,jend,z,h,n,dx,flux)
906 implicit none

```



```

907 real(8), parameter :: hm=5.0d0/3.0d0
908 integer :: ni , n_1d , ij_1d(n_1d,8) , i , j , iend , jend , iofs1 , jofs1 , iofs2 , jofs2 , iofs3 , jofs3
909 real(8) :: z(iend , jend) , h(iend , jend) , wl(iend , jend) , n(iend , jend) , &
910         flux(iend+1 , jend+1) , grad(iend+1 , jend+1) , cost , sint , dx
911 !
912 ! west h(i , j) , (wl(i+1 , j) - wl(i , j)) / dx (0 , 0) , (1 , 0) , (0 , 0) , flux < 0
913 ! east h(i-1 , j) , (wl(i-1 , j) - wl(i-2 , j)) / dx (-1 , 0) , (-1 , 0) , (-2 , 0) , flux > 0
914 ! north h(i , j-1) , (wl(i , j-1) - wl(i , j-2)) / dy (0 , -1) , (0 , -1) , (0 , -2) , flux > 0
915 ! south h(i , j) , (wl(i , j+1) - wl(i , j)) / dy (0 , 0) , (0 , 1) , (0 , 0) , flux < 0
916 wl=z+h
917 do ni=1 , n_1d
918     i =ij_1d(ni , 1)
919     j =ij_1d(ni , 2)
920     iofs1=ij_1d(ni , 3)
921     jofs1=ij_1d(ni , 4)
922     iofs2=ij_1d(ni , 5)
923     jofs2=ij_1d(ni , 6)
924     iofs3=ij_1d(ni , 7)
925     jofs3=ij_1d(ni , 8)
926     grad(i , j)=(wl(i+iofs2 , j+jofs2)-wl(i+iofs3 , j+jofs3))/dx
927     cost=cos(atan(grad(i , j)))
928     sint=sin(atan(grad(i , j)))
929     flux(i , j)=sign(1.0d0/n(i+iofs1 , j+jofs1)) *&
930     abs(sint)**0.50d0*(h(i+iofs1 , j+jofs1)*cost)**hm , -grad(i , j))
931     if(iofs2>0 .or. jofs2>0)then
932         if(flux(i , j)>0.0d0)flux(i , j)=0.0d0
933     else if(iofs2<0 .or. jofs2<0)then
934         if(flux(i , j)<0.0d0)flux(i , j)=0.0d0
935     end if
936 end do
937 !
938 end subroutine cal_q_bd
939 !
940 !
941 !
942 subroutine cal_qs(n_1d , ij_1d , &
943                 iend , jend , z , h , ioffset , joffset , kk , hthr , dx , flux , uvmax , h_uvmax)
944 use omp_lib
945 implicit none
946 integer :: ni , n_1d , ij_1d(1:n_1d , 1:2) , i , j , iend , jend , ip , jp , ioffset , joffset
947 real(8) :: z(iend , jend) , h(iend , jend) , wl(iend , jend) , kk(iend , jend) , &
948         flux(iend+1 , jend+1) , uvmax(iend+1 , jend+1) , h_uvmax(iend+1 , jend+1) , &
949         grad(iend+1 , jend+1) , cost , sint , hthr , dx , htmp , uv
950 !
951 !uvmax=0.d0
952 wl=z+h
953 !$omp parallel do private(ni , i , j , ip , jp , cost , sint , htmp , uv)
954 do ni=1 , n_1d
955     i=ij_1d(ni , 1)
956     j=ij_1d(ni , 2)
957     ip=i+ioffset
958     jp=j+joffset
959     grad(i , j)=(wl(i , j)-wl(ip , jp))/dx
960     cost=cos(atan(grad(i , j)))
961     sint=sin(atan(grad(i , j)))

```

```

962  if ( grad(i , j) > 0.0d0 ) then
963    htmp=wl(i , j) -max(z(i , j),z(ip , jp))
964    htmp=htmp*cost
965    flux(i , j)=sign(abs(sint)*(wl(i , j)-max(z(i , j),z(ip , jp))))*kk(i , j),-grad(i , j))
966    if(htmp>0.d0)then
967      uv=abs(flux(i , j))/htmp
968    else
969      uv=0.d0
970    end if
971    if ( wl(i , j) < z(ip , jp) .or. h(i , j) < hthr ) then
972      flux(i , j) = 0.0d0
973      uv = 0.d0
974    end if
975  else
976    htmp=wl(ip , jp)-max(z(i , j),z(ip , jp))
977    htmp=htmp*cost
978    flux(i , j)=sign(abs(sint)*(wl(ip , jp)-max(z(i , j),z(ip , jp))))*kk(ip , jp), &
979    -grad(i , j))
980    if(htmp>0.d0)then
981      uv=abs(flux(i , j))/htmp
982    else
983      uv=0.d0
984    end if
985    if ( wl(ip , jp) < z(i , j) .or. h(ip , jp) < hthr )then
986      flux(i , j) = 0.0d0
987      uv = 0.d0
988    end if
989  end if
990  if(uvmax(i , j)<uv)then
991    uvmax(i , j)=uv
992    h_uvmax(i , j)=htmp
993  end if
994 end do
995 !$omp end parallel do
996 !call cal_qs_bd(n_1d_bd , ij_1d_bd , iend , jend , z , h , kk , dd , flux)
997 !
998 end subroutine cal_qs
999 !
1000 !
1001 !
1002 subroutine cal_qs_bd(n_1d , ij_1d , iend , jend , z , h , kk , dx , flux)
1003 implicit none
1004 integer :: ni , n_1d , ij_1d(n_1d , 8) , i , j , iend , jend , iofs1 , jofs1 , iofs2 , jofs2 , iofs3 , jofs3
1005 real(8) :: z(iend , jend) , h(iend , jend) , wl(iend , jend) , kk(iend , jend) , &
1006 flux(iend+1 , jend+1) , grad(iend+1 , jend+1) , cost , sint , dx
1007 !
1008 ! west h(i , j) , (wl(i+1 , j)-wl(i , j))/dx ( 0 , 0) , ( 1 , 0) , ( 0 , 0) , flux<0
1009 ! east h(i-1 , j) , (wl(i-1 , j)-wl(i-2 , j))/dx (-1 , 0) , (-1 , 0) , (-2 , 0) , flux>0
1010 ! north h(i , j-1) , (wl(i , j-1)-wl(i , j-2))/dy ( 0 , -1) , ( 0 , -1) , ( 0 , -2) , flux>0
1011 ! south h(i , j) , (wl(i , j+1)-wl(i , j))/dy ( 0 , 0) , ( 0 , 1) , ( 0 , 0) , flux<0
1012 wl=z+h
1013 do ni=1 , n_1d
1014   i =ij_1d(ni , 1)
1015   j =ij_1d(ni , 2)
1016   iofs1=ij_1d(ni , 3)

```

```

1017  jofs1=ij_1d(ni,4)
1018  iofs2=ij_1d(ni,5)
1019  jofs2=ij_1d(ni,6)
1020  iofs3=ij_1d(ni,7)
1021  jofs3=ij_1d(ni,8)
1022  grad(i,j)=(wl(i+iofs2,j+jofs2)-wl(i+iofs3,j+jofs3))/dx
1023  cost=cos(atan(grad(i,j)))
1024  sint=sin(atan(grad(i,j)))
1025  flux(i,j)=sign(abs(sint)*(h(i+iofs1,j+jofs1)*cost)*kk(i+iofs1,j+jofs1), &
1026  -grad(i,j))
1027  if(iofs2>0 .or. jofs2>0)then
1028    if(flux(i,j)>0.0d0)flux(i,j)=0.0d0
1029  else if(iofs2<0 .or. jofs2<0)then
1030    if(flux(i,j)<0.0d0)flux(i,j)=0.0d0
1031  end if
1032 end do
1033 !
1034 end subroutine cal_qs_bd
1035 !
1036 !
1037 !
1038 subroutine cal_divha(n_1d,ij_1d,iend,jend,lamda,pwc,ha,h,hs,pw,d,finf2,dt,qx,qy,qsx,qsy)
1039 use omp_lib
1040 implicit none
1041 integer :: ni,n_1d,ij_1d(n_1d,2),i,j,iend,jend
1042 real(8) :: lamda(iend,jend),pwc(iend,jend),ho, &
1043         ha(iend,jend),h(iend,jend),hs(iend,jend),pw(iend,jend),d(iend,jend), &
1044         finf2(iend,jend),dt,qx(iend+1,jend+1),qy(iend+1,jend+1), &
1045         qsx(iend+1,jend+1),qsy(iend+1,jend+1)
1046 !
1047 !$omp parallel do private(ni,i,j,ho)
1048 do ni=1,n_1d
1049   i=ij_1d(ni,1)
1050   j=ij_1d(ni,2)
1051   ho=h(i,j)
1052   if(ha(i,j)>lamda(i,j)*d(i,j)+finf2(i,j)*dt)then
1053     h(i,j)=ha(i,j)-(lamda(i,j)*d(i,j)+finf2(i,j)*dt)
1054     hs(i,j)=d(i,j)
1055     pw(i,j)=pwc(i,j)
1056   else if(ha(i,j)>pwc(i,j)*d(i,j)+finf2(i,j)*dt)then
1057     h(i,j)=0.d0
1058     hs(i,j)=(ha(i,j)-(pwc(i,j)*d(i,j)+finf2(i,j)*dt))/(lamda(i,j)-pwc(i,j))
1059     pw(i,j)=pwc(i,j)
1060   else if(ha(i,j)>pwc(i,j)*d(i,j))then
1061     h(i,j)=0.d0
1062     hs(i,j)=0.d0
1063     pw(i,j)=pwc(i,j)
1064     finf2(i,j)=(ha(i,j)-pwc(i,j)*d(i,j))/dt
1065   else if(ha(i,j)>=0.d0)then
1066     h(i,j)=0.d0
1067     hs(i,j)=0.d0
1068     pw(i,j)=ha(i,j)/d(i,j)
1069     finf2(i,j)=0.d0
1070   else
1071     write(*,'(a)')'ha<_0,_'@cal_divha->_stop'

```

```

1072 write(*,'(2i5,2f10.3)')i,j,ha(i,j),ho
1073 write(*,'(5f10.3)')qx(i+1,j),qx(i,j),qy(i,j+1),qy(i,j), &
1074 & (-(qx(i+1,j)-qx(i,j))-(qy(i,j+1)-qy(i,j)))*dt
1075 write(*,'(5f10.3)')qsx(i+1,j),qsx(i,j),qsy(i,j+1),qsy(i,j), &
1076 & (-(qsx(i+1,j)-qsx(i,j))-(qsy(i,j+1)-qsy(i,j)))*dt
1077 if(ha(i,j)<=-0.01d0)then
1078 read(*,*)
1079 end if
1080 h(i,j)=0.d0
1081 ha(i,j)=0.d0
1082 !read(*,*)
1083 !stop
1084 end if
1085 end do
1086 !$omp end parallel do
1087 !
1088 end subroutine cal_divha
1089 !
1090 !
1091 !
1092 subroutine cal_eva(n_1d,ij_1d,iend,jend,r,hs,pw,lamda,pwc,d,dt,eva1,eva2)
1093 use omp_lib
1094 implicit none
1095 integer :: ni,n_1d,ij_1d(n_1d,2),i,j,iend,jend
1096 real(8) :: lamda(i,j),pwc(i,j),dt,d(iend,jend), &
1097 r(iend,jend),hs(iend,jend),pw(iend,jend),eva1(iend,jend),eva2(iend,jend)
1098 !
1099 !$omp parallel do private(ni,i,j)
1100 do ni=1,n_1d
1101 i=ij_1d(ni,1)
1102 j=ij_1d(ni,2)
1103 if(r(i,j)>0.d0)then
1104 eva1(i,j)=0.d0
1105 eva2(i,j)=0.d0
1106 else if(hs(i,j)>0.d0)then
1107 eva1(i,j)=min(eva1(i,j),(lamda(i,j)-pwc(i,j))*hs(i,j)/dt)
1108 hs(i,j)=hs(i,j)-eva1(i,j)*dt/(lamda(i,j)-pwc(i,j))
1109 eva2(i,j)=0.d0
1110 else if(pw(i,j)>=0.d0)then
1111 eva1(i,j)=0.d0
1112 eva2(i,j)=min(eva2(i,j),pw(i,j)*d(i,j)/dt)
1113 pw(i,j)=pw(i,j)-eva2(i,j)*dt/d(i,j)
1114 end if
1115 end do
1116 !$omp end parallel do
1117 !
1118 end subroutine cal_eva
1119 !
1120 !
1121 !
1122 subroutine w_fasc(fon,fname,cfmt,a,iend,jend,xllcorner,yllcorner,cellsize,nv)
1123 implicit none
1124 integer :: j,iend,jend,fon
1125 real(8) :: a(iend,jend),xllcorner,yllcorner,cellsize,nv
1126 character(len=100) :: cfmt,fname

```

```

1127 !
1128 open(fon , file=trim(adjustl(fname)))
1129 write(fon , '(a,i5)')'ncols' ,iend
1130 write(fon , '(a,i5)')'nrows' ,jend
1131 write(fon , '(a,f15.3)')'xllcorner' ,xllcorner
1132 write(fon , '(a,f15.3)')'yllcorner' ,yllcorner
1133 write(fon , '(a,f10.3)')'cellsize_' ,cellsize
1134 write(fon , '(a,f10.3)')'NODATA_value' ,nv
1135 do j=jend,1,-1
1136     write(fon ,trim(cfmt))a(1:iend , j)
1137 end do
1138 close(fon)
1139 !
1140 end subroutine w_fasc
1141 !
1142 !
1143 !
1144 subroutine w_iasc(fon ,fname ,ia ,iend ,jend ,xllcorner ,yllcorner ,cellsize ,nv)
1145 implicit none
1146 integer :: j ,iend ,jend ,fon ,nv ,ia(iend ,jend)
1147 real(8) :: xllcorner ,yllcorner ,cellsize
1148 character(len=100) :: fname
1149 !
1150 open(fon , file=trim(adjustl(fname)))
1151 write(fon , '(a,i5)')'ncols' ,iend
1152 write(fon , '(a,i5)')'nrows' ,jend
1153 write(fon , '(a,f15.3)')'xllcorner' ,xllcorner
1154 write(fon , '(a,f15.3)')'yllcorner' ,yllcorner
1155 write(fon , '(a,f10.3)')'cellsize_' ,cellsize
1156 write(fon , '(a,i5)')'NODATA_value' ,nv
1157 do j=jend,1,-1
1158     write(fon , '(*(i2)')ia(1:iend , j)
1159 end do
1160 close(fon)
1161 !
1162 end subroutine w_iasc
1163 !
1164 !
1165 !
1166 subroutine r_iasc(fon ,fname ,ia ,iend ,jend ,xllcorner ,yllcorner ,cellsize)
1167 implicit none
1168 integer :: j ,iend ,jend ,fon ,ncols ,nrows ,ia(iend ,jend)
1169 real(8) :: xllcorner ,yllcorner ,cellsize
1170 character(len=100) :: ctmp ,fname
1171 !
1172 open(fon , file=trim(adjustl(fname)))
1173 read(fon ,*)ctmp ,ncols
1174 read(fon ,*)ctmp ,nrows
1175 if (ncols /= iend .or. nrows /= jend) then
1176     write(* ,*)'Check_ncols_/=_iend_/or_/nrows_/=_jend'
1177     stop
1178 end if
1179 read(fon ,*)ctmp ,xllcorner
1180 read(fon ,*)ctmp ,yllcorner
1181 read(fon ,*)ctmp ,cellsize

```

```

1182 read(fon,*)
1183 do j=jend,1,-1
1184   read(fon,*)ia(1:iend,j)
1185 end do
1186 close(fon)
1187 !
1188 end subroutine r_iasc
1189 !
1190 !
1191 !
1192 subroutine r_fasc(fon,fname,a,iend,jend,xllcorner,yllcorner,cellsize)
1193 implicit none
1194 integer :: j,iend,jend,fon,ncols,nrows
1195 real(8) :: xllcorner,yllcorner,cellsize,a(iend,jend)
1196 character(len=100) :: ctmp,fname
1197 !
1198 open(fon,file=trim(adjustl(fname)))
1199 read(fon,*)ctmp,ncols
1200 read(fon,*)ctmp,nrows
1201 if (ncols /= iend .or. nrows /= jend) then
1202   write(*,*)'Check_ncols/=iend_or_nrows/=jend'
1203   stop
1204 end if
1205 read(fon,*)ctmp,xllcorner
1206 read(fon,*)ctmp,yllcorner
1207 read(fon,*)ctmp,cellsize
1208 read(fon,*)
1209 do j=jend,1,-1
1210   read(fon,*)a(1:iend,j)
1211 end do
1212 close(fon)
1213 !
1214 end subroutine r_fasc
1215 !
1216 !
1217 !
1218 subroutine point(fon,fname,a,iend,jend,xllcorner,yllcorner,cellsize)
1219 implicit none
1220 integer :: i,j,iend,jend,fon,a(iend,jend),iflg
1221 real(8) :: xllcorner,yllcorner,cellsize,x,y
1222 character(len=100) :: fname
1223 !
1224 open(fon,file=trim(adjustl(fname)))
1225 write(fon,'(a)')'x,y,i,j,flag'
1226 do i=1,iend
1227   do j=1,jend
1228     x=xllcorner+cellsize*float(i-1)
1229     y=yllcorner+cellsize*float(j-1)
1230     iflg=int(a(i,j))
1231     write(fon,'(2(f15.3,a),3(i5,a))')x,',',y,',',i,',',j,',',iflg
1232   end do
1233 end do
1234 close(fon)
1235 !
1236 end subroutine point

```

```

1237 !
1238 !
1239 !
1240 subroutine point2(fon,fname,xllcorner,yllcorner,cellsize,n_1d,i_1d,j_1d)
1241 implicit none
1242 integer :: ni,n_1d,i_1d(1:n_1d),j_1d(1:n_1d),i,j,fon
1243 real(8) :: xllcorner,yllcorner,cellsize
1244 character(len=100) :: fname
1245 !
1246 open(fon,file=trim(adjustl(fname)))
1247 write(fon,'(a)')'x,y,i,j'
1248 do ni=1,n_1d
1249   i=i_1d(ni)
1250   j=j_1d(ni)
1251   write(fon,'(2(f15.3,a),2(i5,a))')&
1252     xllcorner+cellsize*float(i-1),',',yllcorner+cellsize*float(j-1),',',i,',',j
1253 end do
1254 close(fon)
1255 !
1256 end subroutine point2
1257 !
1258 !
1259 !
1260 subroutine cal_dq(n_1d,ij_1d,iend,jend,dx,h,qx,qy,dq,dt)
1261 use omp_lib
1262 implicit none
1263 integer :: ni,n_1d,ij_1d(n_1d,2),i,j,iend,jend
1264 real(8) :: dx,dt,h(iend,jend),dq(iend,jend), &
1265           qx(iend+1,jend+1),qy(iend+1,jend+1)
1266 !
1267 !$omp parallel do private(ni,i,j)
1268 do ni=1,n_1d
1269   i=ij_1d(ni,1)
1270   j=ij_1d(ni,2)
1271   dq(i,j)=-((qx(i+1,j)-qx(i,j))/dx+(qy(i,j+1)-qy(i,j))/dx)
1272   if(h(i,j)+dq(i,j)*dt < 0.d0 .and. 0.d0 < dt)then
1273     dt=min(dt,h(i,j)/dq(i,j))
1274   end if
1275 end do
1276 !$omp end parallel do
1277 !
1278 end subroutine cal_dq
1279 !
1280 !
1281 !
1282 subroutine cal_1d_uhb(ni_vect,i_vect,iend,i_to,z,h,b,n,dt,cfl,dx,u,uo,hlim,umax,q)
1283 use omp_lib
1284 implicit none
1285 real(8), parameter :: PI=acos(-1.d0),deg2rad=PI/180.d0,rad2deg=180.d0/PI, &
1286                   hm=5.d0/3.d0,rm=4.d0/3.d0
1287 integer :: i,iend,j,ni_vect,i_vect(ni_vect),i_to(iend)
1288 real(8) :: hlim,u_tmp,h1,r1,b1,umax, cost,sint,dt,cfl,grad(iend),dx(iend), &
1289           z(iend),h(iend),u(iend),uo(iend),n(iend),b(iend),q(0:iend)
1290 !
1291 !           from i to

```

```

1292 ! FX      q      q      q
1293 !          =>    =>    =>
1294 ! CV | z | z | z |
1295 !   |from| i | to |
1296 !umax=0.d0
1297 !$omp parallel do private(j,i,cost,sint,h1,r1,b1,u_tmp) reduction(max: umax)
1298 do j=1,ni_vect
1299   i=i_vect(j)
1300   if(h(i)<hlim .and. h(i_to(i))<hlim)then
1301     u(i)=0.d0
1302     q(i)=0.d0
1303     cycle
1304   end if
1305   grad(i)=(z(i_to(i))+h(i_to(i))-z(i)-h(i))/dx(i)
1306   cost=cos(atan(grad(i)))
1307   sint=sin(atan(grad(i)))
1308   !if(uo(i)>=0.d0)then ! Be careful when uo=0
1309   if(uo(i)>0.d0 .or. (uo(i)==0.d0 .and. grad(i)<=0.d0 .and. h(i)>=hlim))then
1310     h1=z(i)+h(i)-max(z(i),z(i_to(i)))
1311     h1=h1*cost
1312     r1=h1*b(i)/(2.d0*h1+b(i))
1313     b1=b(i)
1314     if(h1<hlim)then
1315       u(i)=0.d0
1316       q(i)=0.d0
1317     cycle
1318   end if
1319   else if(uo(i)<0.d0 .or. (uo(i)==0.d0 .and. grad(i)>0.d0 .and. h(i_to(i))>=hlim))then
1320     h1=z(i_to(i))+h(i_to(i))-max(z(i),z(i_to(i)))
1321     h1=h1*cost
1322     r1=h1*b(i_to(i))/(2.d0*h1+b(i_to(i)))
1323     b1=b(i_to(i))
1324     if(h1<hlim)then
1325       u(i)=0.d0
1326       q(i)=0.d0
1327     cycle
1328   end if
1329   else
1330     u(i)=0.d0
1331     q(i)=0.d0
1332   cycle
1333   end if
1334   !u_tmp=sign(1.d0/rn*h1**(2.d0/3.d0)*abs(grad(i))**0.5d0,-grad(i))
1335   u_tmp=sign(1.d0/n(i)*r1**(2.d0/3.d0)*abs(sint)**0.5d0,-grad(i))
1336   u(i)=u_tmp
1337   if(uo(i)*u(i) < 0.d0)u(i)=0.d0
1338   if(dx(i)/dt*cfl<abs(u(i)) .and. 0.d0<cfl)then
1339     ! write(*,*)'@u_id cfl ',u(i),dx(i)/dt*cfl
1340     ! write(*,*)'@u_id cfl ',dt,dx(i)*cfl/abs(u(i)),dt-dx(i)*cfl/abs(u(i))
1341     !read(*,*)
1342     u(i)=sign(dx(i)/dt*cfl,u(i))
1343   end if
1344   q(i)=b1*h1*u(i) ! A*V
1345   if(abs(u(i))>umax)umax=abs(u(i))
1346 end do

```



```

1347 !$omp end parallel do
1348 !
1349 end subroutine cal_1d_uhb
1350 !
1351 !
1352 !
1353 subroutine cal_1d_h(ni_scal , i_scal , iend , dt , dx , b , h , iqnum , q_in , q_out , dtmin_h1d)
1354 use omp_lib
1355 implicit none
1356 real(8),parameter :: dtmin0=0.00001d0
1357 integer :: i , iend , j , ni_scal , i_scal(ni_scal) , iqnum
1358 real(8) :: dt , dt2 , dtmin_h1d , b(iend) , h(iend) , ho , dx(iend) , q_in(iqnum) , q_out(iqnum)
1359 !
1360 !      from i      to
1361 ! FX      q      q      q
1362 !      => => =>
1363 ! CV | z | z | z |
1364 ! |from| i | to |
1365 dt2=dt
1366 !$omp parallel do private(j,i,ho) reduction(min: dt2,dtmin_h1d)
1367 do j=1,ni_scal
1368   i=i_scal(j)
1369   ho=h(i)
1370   h(i)=ho-(q_out(i)-q_in(i))/dx(i)/b(i)*dt
1371   if(h(i)<0.d0)then
1372     if(dtmin_h1d<=0.d0)then
1373       write(*,'(a,4f15.7)')'h<0,@h_1d_(dt,h,qo,qi)=' ,dt,h(i),q_out(i),q_in(i)
1374       !h(i)=0.d0
1375       !q_out(i)=0.d0
1376       read(*,*)
1377     end if
1378     !write(*,'(a,2i5,5f10.5)')'@1dh h1d < 0 ',i,j,ho,h(i),q_out(i),q_in(i)
1379     !write(*,'(a,f10.5)')'dt=',dt
1380     !read(*,*)
1381     !h(i)=0.d0
1382     !q_out(i)=0.d0
1383     if(0.d0 < (q_out(i)-q_in(i)))then
1384       dt2=min(dt2,ho/(q_out(i)-q_in(i))*dx(i)*b(i))
1385     else
1386       ! write(*,*)'qout-qin=0'
1387       ! read(*,*)
1388       !dt2=dt
1389     end if
1390     !write(*,'(a,3f15.7)')'@h_1d (dt2,h2,dt)=' ,dt2,h(i),dt
1391     if(dt2<0.00001d0)then
1392       write(*,'(a,3f15.7)')'dt<lim@h_1d_(dt2,h2,dt)=' ,dt2,h(i),dt
1393       write(*,*)'stop'
1394       read(*,*)
1395     end if
1396     dtmin_h1d=dt2
1397   end if
1398 end do
1399 !$omp end parallel do
1400 !
1401 end subroutine cal_1d_h

```

```

1402 !
1403 !
1404 !
1405 subroutine cal_2d1d(iend ,jend ,ij_2d1d ,iend_1d ,dx ,dtdiv ,z ,h ,z_1d ,h_1d ,b_1d ,dx_scal ,vsl_out )
1406 !https://www.mlit.go.jp/river/shishin-guideline/gijutsu/gijutsukijunn/chousa/pdf/07.pdf
1407 use omp_lib
1408 implicit none
1409 real(8) , parameter :: u1=2.d0/3.d0**(3.d0/2.d0) ,u2=0.35d0 ,u3=0.91d0 ,g=9.8d0
1410 integer :: i ,j ,k ,iend ,jend ,iend_1d ,ij_2d1d(iend_1d ,2)
1411 real(8) :: dx ,dtdiv ,wl2d ,wl1d ,h1 ,h2 ,qsr ,vsr ,z(iend ,jend) ,h(iend ,jend) , &
1412         z_1d(iend_1d) ,h_1d(iend_1d) ,b_1d(iend_1d) ,dx_scal(iend_1d) ,vsl_out(iend_1d)
1413 !
1414 !$omp parallel do private(k ,i ,j ,wl2d ,wl1d ,h1 ,h2 ,qsr ,vsr)
1415 do k=1,iend_1d
1416     i=ij_2d1d(k,1)
1417     j=ij_2d1d(k,2)
1418     wl2d=z(i ,j)+h(i ,j)
1419     wl1d=z_1d(k)+h_1d(k)
1420     if(wl1d <= wl2d)then ! s->r from surface
1421         h1=h(i ,j)
1422         h2=max(0.d0 ,wl1d-z(i ,j))
1423         !if(z(i ,j) <= wl1d)then
1424             if(h2<2.d0/3.d0*h1)then
1425                 qsr=u2*h1*(2.d0*g*h1)**0.5d0
1426             else
1427                 qsr=u3*h2*(2.d0*g*(h1-h2))**0.5d0
1428             end if
1429         !else ! fall
1430         ! qsr=u1*h1*(g*h1)**0.5d0
1431         !end if
1432     else ! r->s from surface
1433         h1=wl1d-z(i ,j)
1434         h2=h(i ,j)
1435         if(h2/h1<2.d0/3.d0)then
1436             qsr=-u2*h1*(2.d0*g*h1)**0.5d0
1437         else
1438             qsr=-u3*h2*(2.d0*g*(h1-h2))**0.5d0
1439         end if
1440     end if
1441     vsr=qsr*dx_scal(k)*dtdiv*2.d0
1442     if(vsr>=0.d0)then
1443         vsr=min(vsr ,h(i ,j)*(dx*dx))
1444     else
1445         vsr=max(vsr ,-(wl1d-z(i ,j))*(b_1d(k)*dx_scal(k)))
1446     end if
1447     vsl_out(k)=vsl_out(k)+vsr
1448     h_1d(k)=h_1d(k)+vsr/(b_1d(k)*dx_scal(k))
1449     h(i ,j)=h(i ,j)-vsr/(dx*dx)
1450     if(h(i ,j)<0.d0)then
1451         !write(* ,*)'@2d1d h2d<0',h(i ,j) ,i ,j
1452         !read(* ,*)
1453         h(i ,j)=0.d0
1454     end if
1455     if(h_1d(k)<0.d0)then
1456         !write(* ,*)'@2d1d h1d<0',h_1d(k) ,k

```

```

1457     !read(*,*)
1458     h_1d(k)=0.d0
1459   end if
1460 end do
1461 !$omp end parallel do
1462 !
1463 end subroutine cal_2d1d
1464 !
1465 !
1466 !
1467 subroutine cal_grad(iend,jend,dx,z,grdave)
1468 implicit none
1469 integer :: i,iend,j,jend
1470 real(8) :: dx,z(iend,jend),dzdx(iend,jend),dzdy(iend,jend),grdave(iend,jend)
1471 !
1472 dzdx(2:iend-1,:)=(z(3:iend,:)-z(1:iend-2,:))/(2.d0*dx)
1473 dzdx(1,:)=(z(2,:)-z(1,:))/dx
1474 dzdx(iend,:)=(z(iend,:)-z(iend-1,:))/dx
1475 !
1476 dzdy(:,2:jend-1)=(z(:,3:jend)-z(:,1:jend-2))/(2.d0*dx)
1477 dzdy(:,1)=(z(:,2)-z(:,1))/dx
1478 dzdy(:,jend)=(z(:,jend)-z(:,jend-1))/dx
1479 !
1480 grdave(:,:)=(dzdx**2.d0+dzdy**2.d0)**0.5d0
1481 !
1482 end subroutine cal_grad

```

---

```

1  ! Program: DR_ver.1.0.f90
2  ! Copyright (2021) by Public Works Research Institute (P.W.R.I.)
3  ! License: CC-BY-SA
4  !
5  !           from i   to
6  ! FX       q   q   q
7  !           →   →   →
8  ! CV | z | z | z |
9  !     |from| i | to |
10 !
11 implicit none
12 real(8), parameter :: PI=acos(-1.d0), deg2rad=PI/180.d0, rad2deg=180.d0/PI, &
13           g=9.8d0, dtlim=0.001d0, cfl=0.01d0
14 real(8), allocatable :: zini(:), z(:), zs(:), b(:), h(:), ho(:), um(:), &
15           cc(:), cco(:), cf(:), cfo(:), e(:), rho(:), rhm(:), tant_e(:), n(:), &
16           q_in(:), q_out(:), qcc_in(:), qcc_out(:), qcf_in(:), qcf_out(:), &
17           hs(:), d(:), c(:), pw(:), sf(:), hsc(:)
18 real(8), allocatable :: u(:), uo(:), grad_z(:), grad_w(:), grad_s(:), q(:), qcc(:), qcf(:), &
19           qs(:), qs_in(:), qs_out(:), q_a(:), qc_a(:), qf_a(:)
20 integer :: ni_inlet, ni_outlet
21 integer, allocatable :: i_inlet(:), i_outlet(:)
22 real(8), allocatable :: qin0(:), cin0(:), rin0(:), fsr_in0(:, :), fsr_in(:)
23 real(8) :: qin, cin, rin
24 real(8) :: dt, time, end_time, dout_time, fout_time, qdt
25 integer :: i, iend, k, itsp, itsp_end, itsp_dout, itsp_fout, qnum, itsp_q, i_umax, itmp
26 integer :: ni_scal, ni_vect
27 integer, allocatable :: i_node(:), i_to(:), i_from(:, :), i_scal(:), i_vect(:), ifld(:)
28 real(8), allocatable :: dx_scal(:), dx_vect(:)
29 integer :: iflg_1d
30 real(8) :: ba, d_1d, rn_1d, wid_m, wid_p, wid_min, dep_m, dep_p, dep_min
31 real(8) :: sig, dm, tanp, csta, cstad, pc, pf, rhow, gam, cmin, hmin, hlim, umax, humax, &
32           ks, lam, fs2, cohe, hs_ini
33 real(8) :: cin_sum, cout_sum, dz_sum, q_sum, qcc_sum, qcf_sum, qs_sum, &
34           h_sum, hs_sum, cc_sum, cf_sum, fsr_sum
35 logical, allocatable :: mask(:)
36 character(len=100) :: fname, out_dir, ctmp
37 integer :: itr
38 real(8) :: flmax, time0, time1, dxmin, dt0, dtmin
39 ! out → 2D
40 real(8), allocatable :: z_2d(:), va_out(:), vc_out(:), vf_out(:)
41 integer, allocatable :: iacc(:)
42 integer :: iaccmax, iacc_up
43 !
44 open(1, file='DR_input.txt')
45 read(1,*)
46 read(1,*)iend
47 read(1,*)iacc_up
48 read(1,*)
49 read(1,*)dt
50 read(1,*)dout_time
51 read(1,*)fout_time
52 read(1,*)
53 read(1,*)qnum

```

```

54 read(1,*)qdt
55 !
56 allocate(zini(iend),z(iend),zs(iend),b(iend),h(iend),ho(iend),um(iend), &
57         cc(iend),cco(iend),cf(iend),cfo(iend),e(iend),rho(iend),rhm(iend),tant_e(iend),n(
58         iend), &
59         q_in(iend),q_out(iend),qcc_in(iend),qcc_out(iend),qcf_in(iend),qcf_out(iend), &
60         hs(iend),d(iend),c(iend),pw(iend),sf(iend),hsc(iend))
61 allocate(u(iend),uo(iend),grad_z(iend),grad_w(iend),grad_s(iend),q(0:iend),qcc(0:iend),qcf
62         (0:iend), &
63         qs(0:iend),qs_in(iend),qs_out(iend),q_a(0:iend),qc_a(0:iend),qf_a(0:iend))
64 allocate(qin0(0:qnum),cin0(0:qnum),rin0(0:qnum),fsr_in0(0:qnum,iend),fsr_in(iend))
65 allocate(i_node(iend),i_to(iend),i_from(iend,3),dx_scal(iend),dx_vect(iend))
66 allocate(mask(iend),ifld(iend),iacc(iend),z_2d(iend),va_out(iend),vc_out(iend),vf_out(iend))
67 !
68 !open(10,file='bound.BSR.txt')
69 do i=0,qnum-1 ! i=0: initial
70 ! read(10,*)qin0(i),cin0(i),rin0(i)
71 qin0(i)=0.d0
72 cin0(i)=0.d0
73 rin0(i)=0.d0
74 rin0(i)=rin0(i)/dble(1000*3600)
75 end do
76 qin0(qnum)=qin0(qnum-1)
77 cin0(qnum)=cin0(qnum-1)
78 rin0(qnum)=rin0(qnum-1)
79 !close(10)
80 read(1,*)
81 read(1,*)dm
82 read(1,*)pf
83 read(1,*)ks ! (mm/h)
84 ks=ks/100.0d0 ! (mm/h) -> (m/s)
85 read(1,*)csta
86 read(1,*)rhow
87 read(1,*)sig
88 read(1,*)tanp ! deg
89 tanp=tan(tanp*deg2rad) ! deg -> rad
90 read(1,*)cohe
91 read(1,*)iflg_1d,hs_ini,fname
92 if(iflg_1d/=0)then
93     open(10,file=fname)
94     read(10,*)
95     do i=1,iend
96         read(10,*)itmp,hs(i)
97     end do
98     close(10)
99 else
100     hs(:)=hs_ini
101 end if
102 read(1,*)
103 read(1,*)hlim
104 read(1,*)
105 read(1,*)iflg_1d
106 read(1,*)fname
107 !
108 open(10,file=fname)

```

```

107 read(10,*) i
108 if (i/=iend) then
109 write(*,*) 'Check_iend'
110 end if
111 read(10,*)
112 do i=1,iend
113   read(10,*) i_node(i), i_to(i), i_from(i,1), i_from(i,2), i_from(i,3), &
114   z_2d(i), z(i), zs(i), dx_scal(i), b(i), n(i), iacc(i)
115   dx_vect(i)=dx_scal(i)
116 end do
117 close(10)
118 dxmin=minval(dx_vect(:))
119 !
120 read(1,*) d_1d
121 read(1,*) rn_1d
122 read(1,*) wid_m
123 read(1,*) wid_p
124 read(1,*) wid_min
125 read(1,*) dep_m
126 read(1,*) dep_p
127 read(1,*) dep_min
128 close(1)
129 !
130 if (iflg_1d==0) then
131   write(*,*) 'Stream_config_<--->DR_input.txt'
132   do i=1,iend
133     ba=dxmin*dxmin*float(iacc(i))/1000000.d0
134     b(i)=max(wid_min, wid_m*ba**wid_p)
135     d(i)=max(dep_min, dep_m*ba**dep_p)
136     z(i)=z_2d(i)-d(i)
137     n(i)=rn_1d
138     zs(i)=z(i)-d_1d
139   end do
140 else
141   write(*,*) 'Stream_config_<--->', trim(adjustl(fname))
142 end if
143 zini(:)=z(:)
144 ifld(:)=0
145 !
146 open(20, file='DR_streamChecker.txt')
147 write(20, '(a5, ", ", *(a10, :, ", ")') 'id', 'z_2d', 'z_1d', 'zs_1d', 'dx', 'b', 'n'
148 do i=1,iend
149   write(20, '(i5, ", ", 5(f10.3, :, ", "), *(f10.5, :, ", ")') i_node(i), z_2d(i), z(i), zs(i), dx_scal(i),
150     b(i), n(i)
151 end do
152 close(20)
153 !
154 open(10, file='flowVolume_RRtoDR.txt')
155 read(10,*)
156 read(10,*)
157 do i=0,qnum-1 ! i=0: initial
158   read(10, '(a10,i10,(e12.2))') ctmp, k, fsr_in0(i,:)
159   fsr_in0(i,:)=fsr_in0(i,:)/(b(:)*dx_scal(:))/qdt
160 end do
161 fsr_in0(1:qnum-1,:)=fsr_in0(1:qnum-1,:)-fsr_in0(0:qnum-2,:)

```

```

161 fsr_in0(qnum,:)=fsr_in0(qnum-1,:)
162 where(fsr_in0<0.d0)fsr_in0=0.d0
163 close(10)
164 !do i=0,qnum-1 ! i=0: initial
165 ! write(*,'(f5.1,a,f10.3,a)')dble(i)*qdt/3600.d0,'(h)',&
166 ! sum(fsr_in0(i,:))/dble(size(fsr_in0(i,:)))*(1000.d0*3600.d0),' (mm/h)'
167 !end do
168 !
169 mask=i_from(:,1)/=0
170 ni_scal=count(mask)
171 allocate(i_scal(ni_scal))
172 i_scal(1:ni_scal)=pack(i_node(:),mask)
173 !
174 mask=i_to(:)/=0
175 ni_vect=count(mask)
176 allocate(i_vect(ni_vect))
177 i_vect(1:ni_vect)=pack(i_node(:),mask)
178 !
179 mask=i_from(:,1)==0
180 ni_inlet=count(mask)
181 allocate(i_inlet(ni_inlet))
182 i_inlet(1:ni_inlet)=pack(i_node(:),mask)
183 !
184 mask=i_to(:)==0
185 ni_outlet=count(mask)
186 allocate(i_outlet(ni_outlet))
187 i_outlet(1:ni_outlet)=pack(i_node(:),mask)
188 !
189 out_dir='output_DR/'
190 write(*,*)trim(adjustl(out_dir))
191
192 lam=1.d0-csta
193 fs2=0.d0
194 iaccmax=iacc(iacc_up)
195 c(:)=cohe
196 cstad=csta
197 pc=1.d0-pf
198 hmin=0.01d0
199 cmin=0.001d0
200 gam=1.d0
201 !
202 end_time=dble(qnum-1)*qdt
203 itsp_end=nint(end_time/dt)
204 itsp_dout=nint(dout_time/dt)
205 itsp_fout=nint(fout_time/dt)
206 itsp_q=nint(qdt/dt)
207 !
208 h(:)=0.d0
209 u(:)=0.d0
210 e(:)=0.d0
211 ho(:)=h(:)
212 uo(:)=u(:)
213 cc(:)=0.d0
214 cf(:)=0.d0
215 cco(:)=cc(:)

```

```

216 cfo(:)=cf(:)
217 q(:)=0.d0
218 qcc(:)=0.d0
219 qcf(:)=0.d0
220 dz_sum=0.d0 !sum(b(1:iend-1)*dx*(z(1:iend-1)-zs(1:iend-1))*csta)
221 cin_sum=0.d0
222 cout_sum=0.d0
223 !hs(:)=0.d0
224 pw(:)=0.d0 ! <— dummy
225 d(:)=z(:)-zs(:)
226 !
227 q_sum=0.d0
228 qcc_sum=0.d0
229 qcf_sum=0.d0
230 qs_sum=0.d0
231 h_sum=0.d0
232 hs_sum=0.d0
233 cc_sum=0.d0
234 cf_sum=0.d0
235 fsr_sum=0.d0
236 !
237 q_a(:)=0.d0
238 qc_a(:)=0.d0
239 qf_a(:)=0.d0
240 !
241 umax=0.d0
242 humax=0.d0
243 i_umax=0
244 !
245 va_out(:)=0.d0
246 vc_out(:)=0.d0
247 vf_out(:)=0.d0
248 !
249 ! —> cal start
250 open(200, file='DR_summary.txt')
251 write(200, '(a10,*(a15,:))') 'time(s)', 'q_sum', 'qs_sum', 'h_sum', 'hs_sum', &
252 'qcc_sum', 'qcf_sum', 'cc_sum', 'cf_sum', 'dz_sum', 'fsr_sum'
253 open(201, file='flowVolume_ws_DRtoDF.txt')
254 open(202, file='flowVolume_cs_DRtoDF.txt')
255 open(203, file='flowVolume_fs_DRtoDF.txt')
256 dt0=dt
257 dtmin=dt
258 time=0.d0
259 time0=0.d0
260 write(201, '(a10,"",i10,(E20.10e3,,""))') 'time(s)', idnint(time), va_out(:)
261 write(202, '(a10,"",i10,(E20.10e3,,""))') 'time(s)', idnint(time), vc_out(:)
262 write(203, '(a10,"",i10,(E20.10e3,,""))') 'time(s)', idnint(time), vf_out(:)
263 !
264 do itsp=1,itsp_end
265   time=time0+dt
266   k=itsp/itsp_q
267   qin=qin0(k)+(qin0(k+1)-qin0(k))*(itsp-k*itsp_q)/dble(itsp_q)
268   cin=cin0(k)+(cin0(k+1)-cin0(k))*(itsp-k*itsp_q)/dble(itsp_q)
269   fsr_in(:)=fsr_in0(k,:)+(fsr_in0(k+1,:)-fsr_in0(k,:))*(itsp-k*itsp_q)/dble(itsp_q)
270   rin=rin0(k+1)

```



```

271  !
272  flmax=0.d0
273  call cal_u(ni_vect , i_vect , iend ,3 , i_from , i_to , dtlim , g , n , z , b , h , ho , grad_w , dx_vect , &
274          u , uo , hlim , rho , rhm , sig , dm , csta , tanp , cc , umax , i_umax , humax)
275  flmax=umax
276  !
277  time1=time0
278  itr=1
279  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
280  if ((dxmin*cfl) < flmax)then
281      itr=max(1 , ceiling(flmax/(dxmin*cfl)))
282      dt=max(dtlim , dt0/float(itr))
283      !write(* , 'a , f8.4' , advance='yes') 'dt=' , dt
284      if(dt < dtmin)then
285          dtmin=dt
286          itr=nint(dt0/dt)
287      end if
288  end if
289  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
290  umax=0.d0
291  i_umax=0
292  humax=0.d0
293  do while (itr>0)
294      d(:)=z(:)-zs(:)
295      call cal_hs(iend , i_node , iend , dt , dx_scal , fsr_in , lam , b , hs , iend , qs_in , qs_out)
296      where(hs(:)>d(:))
297          h(:)=h(:)+(hs(:)-d(:))*lam
298          hs(:)=d(:)
299      end where
300      !
301      call cal_sf(ni_scal , i_scal , iend , g , rhow , sig , tanp , lam , hlim , d , grad_z , hs , h , pw , c , sf , hsc)
302      where(1.d0<sf(:) .and. d(:)==hs(:))
303          ifld(:)=1
304      end where
305      do i=1 , ni_inlet
306          k=i_to(i_inlet(i))
307          do while (k/=0)
308              if( ifld(i_from(k,1))==1 )then
309                  ifld(k)=1
310              end if
311              k=i_to(k)
312          end do
313      end do
314      where( ifld==0)e=0.d0
315      !
316      call cal_h(iend , i_node , iend , dt , dx_scal , b , h , iend , q_in , q_out , e , csta , cstad)
317      call cal_cc(iend , i_node , iend , dt , dx_scal , b , h , ho , cc , cco , iend , qcc_in , qcc_out , e , gam , pc)
318      call cal_cf(iend , i_node , iend , dt , dx_scal , b , h , ho , cf , cfo , cc , cco , iend , qcf_in , qcf_out , e , gam ,
319              pf , cstad)
320      call cal_z(iend , i_node , iend , dt , csta , z , zs , e , tant_e)
321      z(i_outlet(1))=zs(i_outlet(1))+(z(i_from(i_outlet(1),1))-zs(i_from(i_outlet(1),1)))
322      !
323      call cal_vout(iend , i_node , iend , dt , dx_scal , b , h , z , z_2d , cc , cf , iacc , iaccmax , &
324              va_out , vc_out , vf_out)
325      !

```

```

325  call cal_qs(ni_vect , i_vect , iend , i_to , b , hs , grad_s , qs , ks)
326  i=i_from(i_outlet(1),1)
327  qs(i_outlet(1))=b(i)*ks*hs(i)*min(1.0d0,abs(grad_s(i)))
328  qs_in(i_scal(:))=qs(i_from(i_scal(:),1))+qs(i_from(i_scal(:),2))+qs(i_from(i_scal(:),3))
329  qs_out(:)=qs(i_node(:))
330  do i=1,ni_inlet
331     qs_in(i_inlet(i))=0.d0
332  end do
333  !
334  call cal_u(ni_vect , i_vect , iend , 3 , i_from , i_to , dt , g , n , z , b , h , ho , grad_w , dx_vect , &
335     u , uo , hlim , rho , rhm , sig , dm , csta , tanp , cc , umax , i_umax , humax)
336  u(i_outlet(1))=u(i_from(i_outlet(1),1))
337  !
338  call cal_q(ni_vect , i_vect , iend , i_to , b , h , u , q(0:iend) , cc , cf , qcc(0:iend) , qcf(0:iend))
339  ! one outlet
340  i=i_outlet(1)
341  q(i)=b(i)*h(i)*u(i)
342  qcc(i)=q(i)*cc(i)
343  qcf(i)=q(i)*cf(i)*(1.d0-cc(i))
344  !
345  q_in(i_scal(:))=q(i_from(i_scal(:),1))+q(i_from(i_scal(:),2))+q(i_from(i_scal(:),3))
346  qcc_in(i_scal(:))=qcc(i_from(i_scal(:),1))+qcc(i_from(i_scal(:),2))+qcc(i_from(i_scal
(:),3))
347  qcf_in(i_scal(:))=qcf(i_from(i_scal(:),1))+qcf(i_from(i_scal(:),2))+qcf(i_from(i_scal
(:),3))
348  !
349  q_out(:)=q(i_node(:))
350  qcc_out(:)=qcc(i_node(:))
351  qcf_out(:)=qcf(i_node(:))
352  !
353  do i=1,ni_inlet
354     q_in(i_inlet(i))=qin
355     qcc_in(i_inlet(i))=qin*pc*cin
356     qcf_in(i_inlet(i))=qin*pf*cin
357  end do
358  !
359  rho(i_node)=(sig-rhow)*cf(:)+rhow
360  rhm(:)=(sig-rho(:))*cc(:)+rho(:)
361  !
362  do k=1,ni_vect
363     i=i_vect(k)
364     grad_w(i)=(z(i_to(i))+h(i_to(i))-z(i)-h(i))/dx_scal(i)
365     grad_s(i)=(zs(i_to(i))+hs(i_to(i))-zs(i)-hs(i))/dx_scal(i)
366     grad_z(i)=(z(i_to(i))-z(i))/dx_scal(i)
367  end do
368  grad_w(i_outlet(1))=grad_w(i_from(i_outlet(1),1))
369  grad_s(i_outlet(1))=grad_s(i_from(i_outlet(1),1))
370  grad_z(i_outlet(1))=grad_z(i_from(i_outlet(1),1))
371  !
372  um(i_scal(:))=(u(i_scal(:))+u(i_from(i_scal(:),1)))*0.5d0
373  where(h(i_inlet(:))<hlim)
374     um(i_inlet(:))=u(i_inlet(:))*0.5d0
375  elsewhere
376     um(i_inlet(:))=(u(i_inlet(:))+qin/(h(i_inlet(:))*b(i_inlet(:))))*0.5d0
377  end where

```

```

378 um(i_inlet(:))=u(i_inlet(:))
379 !
380 call cal_e(iend,i_node,iend,dt,dx_scal,b,z,zs,sig,rho,csta,cstad,pf,pc,tanp,cc,cf,e,h,um
,grad_z, &
381 q_in,q_out,qcc_in,qcc_out,qcf_in,qcf_out,tant_e)
382 !
383 uo(:)=u(:)
384 ho(:)=h(:)
385 cco(:)=cc(:)
386 cfo(:)=cf(:)
387 !
388 q_a(:)=q_a(:)+q(:)*dt
389 qc_a(:)=qc_a(:)+qcc(:)*dt
390 qf_a(:)=qf_a(:)+qcf(:)*dt
391 !
392 cin_sum=cin_sum+sum(qcc_in(i_inlet(:)))*dt+sum(qcf_in(i_inlet(:)))*dt
393 cout_sum=cout_sum+qcc(i_outlet(1))*dt+qcf(i_outlet(1))*dt
394 q_sum=q_sum+q(i_outlet(1))*dt
395 qcc_sum=qcc_sum+qcc(i_outlet(1))*dt
396 qcf_sum=qcf_sum+qcf(i_outlet(1))*dt
397 qs_sum=qs_sum+qs(i_outlet(1))*dt
398 h_sum=sum(h(:)*b(:)*dx_scal(:))
399 hs_sum=sum(hs(:)*b(:)*dx_scal(:))*lam
400 dz_sum=sum(d(:)*b(:)*dx_scal(:))
401 cc_sum=sum(h(:)*cc(:)*b(:)*dx_scal(:))
402 cf_sum=sum(h(:)*(1.d0-cc(:))*cf(:)*b(:)*dx_scal(:))
403 fsr_sum=fsr_sum+sum(fsr_in(:))*dt
404 !
405 time1=time1+dt
406 itr=itr-1
407 end do ! do while
408 !
409 time0=time
410 dt=dt0
411 ! output
412 if (mod(itstp, itstp_dout) ==0) then
413 write(*,'(a,i8,5(a,f10.3))')'time(s)', nint(time), &
414 ',_uqin=',sum(q_in(i_inlet(:))),',_uqout=',q(i_outlet(1)),',_uuuamax=',umax
415 write(*,'(a,f8.3,2(a,f10.3),(a,i6))')'time(h)',(time/3600.d0), &
416 ',_uqcc=',sum(qcc_in(i_inlet(:))),',_uqout=',qcc(i_outlet(1)),',_ui@umax=',i_umax
417 write(*,'(a,5(a,f10.3))')'uuuuuuuuuuuuuuuuuu', &
418 ',_uqcf=',sum(qcf_in(i_inlet(:))),',_uqout=',qcf(i_outlet(1)),',_uh@umax=',humax
419 write(*,'(2(a,f10.5))')'dt_min=',dtmin,',_dt_lim=',dtlim
420 umax=0.d0
421 humax=0.d0
422 dtmin=dt
423 ! —> summary
424 write(200,'(i10,"",*(e15.5e3,:","))') nint(time),q_sum,qs_sum,h_sum,hs_sum, &
425 qcc_sum,qcf_sum,cc_sum,cf_sum,dz_sum,fsr_sum
426 ! <— summary
427 write(201,'(a10,"",i10,(E20.10e3,:","))')'time(s)',idnint(time),va_out(:)
428 write(202,'(a10,"",i10,(E20.10e3,:","))')'time(s)',idnint(time),vc_out(:)
429 write(203,'(a10,"",i10,(E20.10e3,:","))')'time(s)',idnint(time),vf_out(:)
430 end if
431 ! —> output to files

```

```

432  if(mod(itssp , itssp_fout) ==0) then
433    write(fname, '(i10.10)') idnint(time)
434    fname=trim(adjustl(out_dir))// 'res_'//trim(adjustl(fname))// '.txt'
435    open(100, file=fname)
436    write(100, '(a5,15a12,3a15)') 'id', 'z', 'h', 'u', 'q', 'cc', 'cf', 'rho', 'e', &
437    'zini', 'grd_w', 'hs', 'qs', 'sf', 'hsc/d', 'ifld', 'qa', 'qcca', 'qcfa'
438    do i=1,iend
439      write(100, '(i5,14f12.5,i12,3e15.5e3)') i, z(i), h(i), um(i), q(i), cc(i), cf(i), rho(i), e(i),
440      zini(i), &
441      grad_w(i), hs(i), qs(i), sf(i), hsc(i), ifld(i), q_a(i), qc_a(i), qf_a(i)
442    end do
443    close(100)
444  end if
445  !<— output to files
446  end do
447  !<— cal start
448  close(200)
449  close(201)
450  close(202)
451  close(203)
452  write(*,*) '---_nomal_end_---'
453  !
454  stop
455  !
456  !
457  !
458  subroutine cal_u(ni_vect, i_vect, iend, nifrom, i_from, i_to, dt, g, n, z, b, h, ho, grad, &
459  dx, u, uo, hlim, rho, rhm, sig, dm, csta, tanp, c, umax, i_umax, humax)
460  use omp_lib
461  implicit none
462  real(8), parameter :: PI=acos(-1.d0), deg2rad=PI/180.d0, rad2deg=180.d0/PI, &
463  hm=5.d0/3.d0, rm=4.d0/3.d0
464  real(8), parameter :: f1d5=1.d0/5.d0, f1d3=1.d0/3.d0, f5d3=5.d0/3.d0, f2d3=2.d0/3.d0, &
465  kf=0.16d0, kd=0.0828d0, e2=0.85d0**2.d0
466  integer :: i, iend, j, i_umax, iloc(1), ni_vect, i_vect(ni_vect), nifrom, i_from(iend, nifrom), i_to(
467  iend)
468  real(8) :: dt, g, x1, x2, x3, hlim, h(iend), ho(iend), u(iend), uo(iend), &
469  h1, b1, c1, rho1, rhm1, n1, z1, h2, z2, hnw, &
470  c(iend), rho(iend), rhm(iend), z(iend), b(iend), n(iend), &
471  sig, csta, dm, taub, tauy, fb, fb1, fb2, tanp, fd, ff, cost, &
472  grad(iend), dx(iend), umax, uhb(nifrom), humax, htmp(iend)
473  !
474  umax=0.d0
475  u(:)=0.d0
476  htmp(:)=0.d0
477  humax=0.d0
478  i_umax=0
479  !$omp parallel do private(i, j, cost, hnw, h1, z1, h2, z2, b1, n1, c1, rho1, rhm1, &
480  !$omp x1, x2, x3, tauy, fd, ff, fb1, fb2, fb, taub)
481  do j=1, ni_vect
482    i=i_vect(j)
483    grad(i)=sign(min(abs(grad(i)), tanp), grad(i))
484    cost=cot(atan(grad(i)))
485    if(uo(i)>=0.d0)then ! when uo=0

```

```

485     hnw=h(i)
486     if(hnw<=0.d0)then
487         u(i)=0.d0
488         cycle
489     end if
490     h1=ho(i)
491     z1=z(i)
492     h2=ho(i_to(i))
493     z2=z(i_to(i))
494     b1=b(i)
495     n1=n(i)
496     c1=c(i)
497     rho1=rho(i)
498     rhm1=rhm(i)
499     if(h1<hlim .or. z1+h1<z2)then
500         u(i)=0.d0
501         cycle
502     end if
503     where(i_from(i,:)==0)
504         uhb(:)=0.d0
505     elsewhere
506         uhb(:)=uo(i_from(i,:))*ho(i_from(i,:))*b(i_from(i,:))
507     end where
508     x1=uo(i)*(uo(i)*ho(i)*b(i)-sum(uhb(:)))/(dx(i)*b(i))
509 else
510     hnw=h(i_to(i))
511     if(hnw<=0.d0)then
512         u(i)=0.d0
513         cycle
514     end if
515     h1=ho(i_to(i))
516     h2=ho(i)
517     z2=z(i)
518     b1=b(i_to(i))
519     n1=n(i_to(i))
520     c1=c(i_to(i))
521     rho1=rho(i_to(i))
522     rhm1=rhm(i_to(i))
523     if(h1<hlim .or. z1+h1<z2)then
524         u(i)=0.d0
525         cycle
526     end if
527     x1=uo(i)*(uo(i_to(i))*h1*b1-uo(i)*ho(i)*b(i))/(dx(i)*b(i))
528 end if
529 !
530 x2=g*h1*grad(i)
531 !
532 if(c1<0.05d0)then
533     !x3=g*n1*n1*uo(i)*abs(uo(i))/h1**f1d3
534     fb=(6.d0+2.5d0*log((h1/dm)))**(-2.d0)
535 else
536     fd=kd*sig/rho1*(1.d0-e2)*c1**(f1d3)
537     ff=kf*(1.d0-c1)**f5d3/c1**(f2d3)
538     fb1=6.25d0*(fd+ff)*((h1/dm))**(-2.d0)
539     fb2=(6.d0+2.5d0*log((h1/dm)))**(-2.d0)

```

```

540     fb=fb1
541     if (fb1<fb2) fb=fb2
542 end if
543 tauy=(c1/csta)**f1d5*(sig-rho1)*c1*g*h1*cost*tanp
544 taub=sign(tauy+rho1*fb*uo(i)*uo(i),uo(i))
545 x3=-taub/rhm1
546 u(i)=(uo(i)*h1-(-X1-X2+tauy/rhm1)*dt)/(hnw+rho1*fb*abs(uo(i))*dt/rhm1)
547 !u(i)=(uo(i)*h1+(x1+x2+x3)*dt)/hnw
548 if (u(i)*uo(i)<0.d0)then
549     u(i)=0.d0
550 end if
551 end do
552 !$omp end parallel do
553 umax=maxval(abs(u(:)))
554 iloc=maxloc(abs(u(:)))
555 i_umax=iloc(1)
556 humax=h(i_umax)
557 !
558 end subroutine cal_u
559 !
560 !
561 !
562 subroutine cal_q(ni_vect,i_vect,iend,i_to,b,h,u,q,cc,cf,qcc,qcf)
563 use omp_lib
564 implicit none
565 integer :: i,iend,j,ni_vect,i_vect(ni_vect),i_to(iend)
566 real(8) :: b(iend),h(iend),u(iend),q(0:iend),cc(iend),cf(iend),qcc(0:iend),qcf(0:iend)
567 !
568 !$omp parallel do private(i,j)
569 do j=1,ni_vect
570     i=i_vect(j)
571     if (u(i)>=0.d0)then
572         q(i)=b(i)*h(i)*u(i)
573         qcc(i)=q(i)*cc(i)
574         qcf(i)=q(i)*cf(i)*(1.d0-cc(i))
575     else
576         q(i)=b(i_to(i))*h(i_to(i))*u(i)
577         qcc(i)=q(i)*cc(i_to(i))
578         qcf(i)=q(i)*cf(i_to(i))*(1.d0-cc(i_to(i)))
579     end if
580 end do
581 !$omp end parallel do
582 !
583 end subroutine cal_q
584 !
585 !
586 !
587 subroutine cal_qs(ni_vect,i_vect,iend,i_to,b,hs,grad,qs,ks)
588 use omp_lib
589 implicit none
590 integer :: i,iend,j,ni_vect,i_vect(ni_vect),i_to(iend)
591 real(8) :: b(iend),hs(iend),grad(iend),qs(0:iend),ks,tant
592 !
593 !$omp parallel do private(i,j,tant)
594 do j=1,ni_vect

```

```

595  i=i_vect(j)
596  tant=grad(i)
597  if(tant>0.d0)then
598    qs(i)=sign(b(i_to(i))*ks*hs(i_to(i))*min(1.0d0,abs(tant)),-tant)
599  else
600    qs(i)=sign(b(i)*ks*hs(i)*min(1.0d0,abs(tant)),-tant)
601  end if
602 end do
603 !$omp end parallel do
604 !
605 end subroutine cal_qs
606 !
607 !
608 !
609 subroutine cal_e(ni_scal , i_scal , iend , dt , dx , b , z , zs , sig , rho , csta , cstad , pf , pc , tanp , cc , cf , e , h , um
      , gradm , &
610                q_in , q_out , qcc_in , qcc_out , qcf_in , qcf_out , tant_e)
611 use omp_lib
612 implicit none
613 real(8) , parameter :: PI=acos(-1.d0) , deg2rad=PI/180.d0 , rad2deg=180.d0/PI
614 integer :: i , iend , j , ni_scal , i_scal(ni_scal)
615 real(8) :: dt , sig , tanp , tant , tane , csta , cstad , pf , pc , emax , emin1 , emin2 , emin3 , &
616          z(iend) , zs(iend) , cc(iend) , cf(iend) , e(iend) , h(iend) , gradm(iend) , &
617          um(iend) , rho(iend) , tant_e(iend) , q_in(iend) , q_out(iend) , &
618          qcc_in(iend) , qcc_out(iend) , qcf_in(iend) , qcf_out(iend) , dx(iend) , b(iend)
619 !
620 !$omp parallel do private(i , j , tant , tane , emax , emin1 , emin2 , emin3)
621 do j=1 , ni_scal
622  i=i_scal(j)
623  if(um(i)>=0.d0)then
624    tant=gradm(i)
625  else
626    tant=gradm(i)
627  end if
628  tant=max(-gradm(i) , 0.d0)
629  tant_e(i)=tant
630  tane=(sig/rho(i)-1.d0)*cc(i)*tanp/((sig/rho(i)-1.d0)*cc(i)+1.d0)
631  ! tan(a-b)=(tan(a)-tan(b))/(1+tan(a)*tan(b))
632  e(i)=csta*(tant-tane)/(1.d0+tant*tane)*abs(um(i))
633  if(e(i)>=0.d0)then ! ero
634    emax=(z(i)-zs(i))/dt*csta*cos(atan(tant))
635    e(i)=min(emax , e(i))
636    ! if(ifld(i)==1)e(i)=emax
637  else ! depo
638    ! -h(i)/dt+(q(i+1)-q(i))/dx=e(i)/csta
639    ! -cc(i)*h(i)/dt+(qcc(i+1)-qcc(i))/dx=e(i)
640    ! -cf(i)*(1.d0-cc(i))*h(i)/dt+(qcf(i+1)-qcf(i))/dx=(1.d0-csta)/csta*cf(i)*e(i)
641    emin1=min(0.d0 , (-h(i)/dt*cstad+(q_out(i)-q_in(i))/dx(i)/b(i))*cstad)
642    emin2=min(0.d0 , -cc(i)*h(i)/dt+(qcc_out(i)-qcc_in(i))/dx(i)/b(i))
643    if(cf(i)>0.d0)then
644      emin3=min(0.d0 , (-cf(i)*(1.d0-cc(i))*h(i)/dt+(qcf_out(i)-qcf_in(i))/dx(i)/b(i))*cstad
        /((1.d0-cstad)*cf(i)))
645    else
646      emin3=e(i)
647  end if

```

```

648     e(i)=max(emin1,emin2,emin3,e(i))
649   end if
650 end do
651 !$omp end parallel do
652 !
653 end subroutine cal_e
654 !
655 !
656 !
657 subroutine cal_h(ni_scal,i_scal,iend,dt,dx,b,h,iqnum,q_in,q_out,e,csta,cstad)
658 use omp_lib
659 implicit none
660 integer :: i,iend,j,ni_scal,i_scal(ni_scal),iqnum
661 real(8) :: dt,csta,cstad,b(iend),h(iend),e(iend),dx(iend),q_in(iqnum),q_out(iqnum)
662 !
663 !$omp parallel do private(i,j)
664 do j=1,ni_scal
665   i=i_scal(j)
666   if(e(i)>=0.d0)then
667     h(i)=h(i)-(q_out(i)-q_in(i))/dx(i)/b(i)*dt+e(i)/csta*dt
668   else
669     h(i)=h(i)-(q_out(i)-q_in(i))/dx(i)/b(i)*dt+e(i)/cstad*dt
670   end if
671   if(h(i)<0.d0)then
672     if(h(i)<-0.001d0)then
673       write(*,'(a,f10.3,i5,f12.7)')'h<0 @ (i)',h(i),i,e(i)
674       write(*,*)q_out(i),q_in(i)
675       !read(*,*)
676     end if
677     h(i)=0.d0
678   end if
679 end do
680 !$omp end parallel do
681 !
682 end subroutine cal_h
683 !
684 !
685 !
686 subroutine cal_hs(ni_scal,i_scal,iend,dt,dx,rin,lam,b,hs,iqnum,qs_in,qs_out)
687 use omp_lib
688 implicit none
689 integer :: i,iend,j,ni_scal,i_scal(ni_scal),iqnum
690 real(8) :: dt,lam,b(iend),hs(iend),rin(iend),dx(iend),qs_in(iqnum),qs_out(iqnum)
691 !
692 !$omp parallel do private(i,j)
693 do j=1,ni_scal
694   i=i_scal(j)
695   hs(i)=hs(i)-(qs_out(i)-qs_in(i))/dx(i)/b(i)/lam*dt+rin(i)/lam*dt
696   if(hs(i)<0.d0)then
697     !write(*,'(a,i5,3f12.7)')'hs<0 @ (i)',i,hs(i),qs_out(i),qs_in(i)
698     !read(*,*)
699   end if
700 end do
701 !$omp end parallel do
702 !

```



```

703 end subroutine cal_hs
704 !
705 !
706 !
707 subroutine cal_sf(ni_scal, i_scal, iend, g, rhow, sig, tanp, lam, hlim, d, grad, hs, h, pw, c, sf, hsc)
708 use omp_lib
709 implicit none
710 integer :: i, iend, j, ni_scal, i_scal(ni_scal)
711 real(8) :: g, rhow, sig, lam, hlim, tant, cost, sint, tanp, ctmp, fg, fr, hsc0, &
712          d(iend), grad(iend), hs(iend), h(iend), pw(iend), c(iend), sf(iend), hsc(iend)
713 !
714 !$omp parallel do private(i, j, tant, cost, sint, fg, fr, hsc0)
715 do j=1, ni_scal
716   i=i_scal(j)
717   tant=grad(i)
718   cost=cos(atan(tant))
719   sint=sin(atan(tant))
720   if(d(i)>=hlim)then
721     ctmp=c(i)/(rhow*g*d(i)*cost*tanp)
722     fg=rhow*g*d(i)*sint*(sig/rhow*(1.d0-lam)+(1.d0-hs(i)/d(i))*pw(i)+hs(i)/d(i)*lam+h(i)/d(i)
723     )
724     fr=rhow*g*d(i)*cost*(sig/rhow*(1.d0-lam)+(1.d0-hs(i)/d(i))*pw(i)-hs(i)/d(i)*(1.d0-lam))*
725     tanp+c(i)
726     sf(i)=fg/fr
727     ! if(sf(i)>1.d0) if l d(i)=1
728     hsc0=((1.d0-tant/tanp)*((1.d0-lam)*sig/rhow+pw(i))+ctmp) &
729     /(((1.d0-tant/tanp)*((1.d0-lam)+pw(i))+tant/tanp)
730     hsc(i)=max(0.d0, min(1.d0, hsc0/d(i)))
731   else
732     sf(i)=1.d0
733     hsc(i)=1.d0
734   end if
735 end do
736 !$omp end parallel do
737 !
738 !
739 !
740 subroutine cal_cc(ni_scal, i_scal, iend, dt, dx, b, h, ho, cc, cco, iqnum, q_in, q_out, e, gam, pc)
741 use omp_lib
742 implicit none
743 integer :: i, iend, j, ni_scal, i_scal(ni_scal), iqnum
744 real(8) :: dt, gam, pc, b(iend), h(iend), ho(iend), cc(iend), cco(iend), e(iend), &
745          dx(iend), q_in(iqnum), q_out(iqnum)
746 !
747 !$omp parallel do private(i, j)
748 do j=1, ni_scal
749   i=i_scal(j)
750   if(h(i)<=0.d0)then ! checking h(i) == hmin
751     cc(i)=0.d0
752   else
753     if(e(i)>=0.d0)then
754       cc(i)=(cco(i)*ho(i)-(q_out(i)-q_in(i))*gam/dx(i)/b(i)*dt+pc*e(i)*dt)/h(i)
755     else

```

```

756     cc(i)=(cco(i)*ho(i)-(q_out(i)-q_in(i))*gam/dx(i)/b(i)*dt+e(i)*dt)/h(i)
757     end if
758 end if
759 if(cc(i)<0.d0)then
760     if(cc(i)*h(i)<-0.0001d0)then
761         write(*,'(a,i5,5f15.7)')'cc*h<0_@_i,cc,h,e',i,cc(i),h(i),e(i)*dt/h(i)
762         !read(*,*)
763     end if
764     cc(i)=0.d0
765 end if
766 end do
767 !$omp end parallel do
768 !
769 end subroutine cal_cc
770 !
771 !
772 !
773 subroutine cal_cf(ni_scal , i_scal , iend , dt , dx , b , h , ho , cf , cfo , cc , cco , iqnum , q_in , q_out , e , gam , pf ,
    cstad)
774 use omp_lib
775 implicit none
776 integer :: i , iend , j , ni_scal , i_scal (ni_scal) , iqnum
777 real(8) :: dt , gam , pf , cstad , b(iend) , h(iend) , ho(iend) , cf(iend) , cfo(iend) , e(iend) , &
778     cc(iend) , cco(iend) , dx(iend) , q_in(iqnum) , q_out(iqnum)
779 !
780 !$omp parallel do private(i , j)
781 do j=1 , ni_scal
782     i=i_scal(j)
783     if(h(i)<=0.d0)then ! checking h(i)===hmin
784         cf(i)=0.d0
785     else
786         if(e(i)>=0.d0)then
787             cf(i)=(cfo(i)*(1.d0-cco(i))*ho(i) &
788                 -(q_out(i)-q_in(i))*gam/dx(i)/b(i)*dt+pf*e(i)*dt) &
789                 /((1.d0-cc(i))*h(i))
790         else
791             cf(i)=(cfo(i)*(1.d0-cco(i))*ho(i) &
792                 -(q_out(i)-q_in(i))*gam/dx(i)/b(i)*dt &
793                 +((1.d0-cstad)/cstad)*cfo(i)*e(i)*dt) &
794                 /((1.d0-cc(i))*h(i))
795         end if
796     end if
797 end do
798 if(cf(i)<0.d0)then
799     if(cf(i)<-0.0001d0)then
800         write(*,'(a,i5,5f15.7)')'cf<0_@_i,cf,h',i,cf(i),h(i)
801         read(*,*)
802     end if
803     cf(i)=0.d0
804 end if
805 end do
806 !$omp end parallel do
807 !
808 end subroutine cal_cf
809 !

```

```

810 !
811 !
812 subroutine cal_z(ni_scal , i_scal , iend , dt , csta , z , zs , e , tant_e)
813 use omp_lib
814 implicit none
815 real(8) , parameter :: PI=acos(-1.d0) , deg2rad=PI/180.d0 , rad2deg=180.d0/PI
816 integer :: i , iend , j , ni_scal , i_scal(ni_scal)
817 real(8) :: dt , csta , tant , cost , z(iend) , zs(iend) , e(iend) , tant_e(iend)
818 !
819 !$omp parallel do private(i , j , tant , cost)
820 do j=1 , ni_scal
821   i=i_scal(j)
822   !tant=slope(i)
823   tant=tant_e(i)
824   cost=cos(atan(tant))
825   z(i)=z(i)-e(i)/csta/cost*dt
826   if(z(i)<zs(i))then
827     !write(* , '(a , i5 , 5f10.3)') 'z < zs at ' , i , z(i) , zs(i) , e(i)
828     !read(* , *)
829     z(i)=zs(i)
830   end if
831 end do
832 !$omp end parallel do
833 !
834 end subroutine cal_z
835 !
836 !
837 !
838 subroutine cal_vout(ni_scal , i_scal , iend , dt , dx , b , h , z1 , z2 , cc , cf , iacc , iaccmax , &
839                   va_out , vc_out , vf_out)
840 use omp_lib
841 implicit none
842 real(8) , parameter :: u2=0.35d0 , g=9.8d0
843 integer :: ni_scal , i_scal(ni_scal) , i , iend , j , iacc(iend) , iaccmax
844 real(8) :: dt , wl , h1 , qout , vtmp , b(iend) , dx(iend) , h(iend) , z1(iend) , z2(iend) , &
845           cc(iend) , cf(iend) , va_out(iend) , vc_out(iend) , vf_out(iend)
846 !
847 !$omp parallel do private(i , j , wl , h1 , qout , vtmp)
848 do j=1 , ni_scal
849   i=i_scal(j)
850   wl=z1(i)+h(i)
851   if(z2(i)<wl .and. iaccmax<iacc(i))then
852     h1=min(h(i) , wl-z2(i))
853     qout=u2*h1*(2.d0*g*h1)**0.5d0
854     vtmp=qout*dt*2.d0*dx(i)
855     va_out(i)=va_out(i)+vtmp
856     vc_out(i)=vc_out(i)+vtmp*cc(i)
857     vf_out(i)=vf_out(i)+vtmp*(1.d0-cc(i))*cf(i)
858     h(i)=h(i)-vtmp/(b(i)*dx(i))
859     if(h(i)<0.d0)then
860       write(* , *)h(i) , wl , z2(i) , z1(i)
861       read(* , *)
862     end if
863   end if
864 end do

```

```
865 !$omp end parallel do  
866 !  
867 end subroutine cal_vout
```

---

## 4.2 02\_DF

DF\_ver.1.0.f90

```

1  ! Program: DF_ver.1.0.f90
2  ! Copyright (2021) by Public Works Research Institute (P.W.R.I.)
3  ! License: CG-BY-SA
4  !-----
5  !   1   2   3   end end+1
6  ! FX  q   q   q   q   q
7  !   →  →  →  →  →
8  ! CV  | z | z | z |...| z |
9  !     | 1 | 2 | 3 |...|end|
10 !-----
11 implicit none
12 real(8), parameter :: PI=acos(-1.d0),deg2rad=PI/180.d0,rad2deg=180.d0/PI, &
13     g=9.8d0,u2=0.35d0,dtlim=0.0001d0,cfl=0.01d0
14 real(8), allocatable :: z(:,,:),zo(:,,:),zini(:,,:),zs(:,,:),h(:,,:),ho(:,,:),d(:,,:), &
15     hs(:,,:),hso(:,,:),coh(:,,:),sf(:,,:),hmax(:,,:),hsmx(:,,:), &
16     cc(:,,:),cco(:,,:),cf(:,,:),cfo(:,,:),e(:,,:),tant_e(:,,:), &
17     grad_z(:,,:),grad_w(:,,:),rho(:,,:),rhom(:,,:),fdir(:,,:)
18 real(8), allocatable :: u(:,,:),uo(:,,:),v(:,,:),vo(:,,:),ua(:,,:),va(:,,:),uv(:,,:), &
19     uh(:,,:),vh(:,,:),qx(:,,:),qy(:,,:),qsx(:,,:),qsy(:,,:), &
20     qccx(:,,:),qccy(:,,:),qcfx(:,,:),qcfy(:,,:), &
21     q_a(:,,:),qcc_a(:,,:),qcf_a(:,,:)
22 real(8), allocatable :: qin0(:),cin0(:)
23 integer, allocatable :: iacc(:,,:),ibasin(:,,:),idir(:,,:),ifld(:,,:)
24 real(8) :: dx,dy,dt,qdt,dout_time,fout_time,xllcorner,yllcorner,cellsize, &
25     dm,csta,cstad,pf,pc,rhow,sig,tanp,coh_tmp,ks,rn,hlim
26 real(8) :: time,time0,time1,end_time,dt0,dtmin, &
27     x,y,flxd,umax,vmax,h_umax,h_vmax,tmp,usmax,vsmax, &
28     qin,cin,q_out,qcc_out,qcf_out
29 integer :: iend,jend,itsp_max,itsp_dout,itsp_fout,qnum,itsp_q
30 integer :: i,j,k,itsp,itr,i_umax,j_umax,i_vmax,j_vmax
31 integer :: ijofs_uv(8,0:4,2)= & ! see subroutine cal_u
32 reshape((/0, 0,-1,-1, 0, 1, 1, 0,& ! @u case0 i=1:4,j=5:8
33     0,-1, 0, 0, 0, 0, 0,-1,& ! @u case1 i=1:4,j=5:8
34     0,-1, 0, 0, 0, 0, 1, 0,& ! @u case2 i=1:4,j=5:8
35     1, 0, 0, 0, 0, 0, 0,-1,& ! @u case3 i=1:4,j=5:8
36     1, 0, 0, 0, 0, 0, 1, 0,& ! @u case4 i=1:4,j=5:8
37     0, 1, 1, 0, 0, 0,-1,-1,& ! @v case0 i=1:4,j=5:8
38     0, 0, 0,-1, 0,-1, 0, 0,& ! @v case1 i=1:4,j=5:8
39     0, 0, 1, 0, 0,-1, 0, 0,& ! @v case2 i=1:4,j=5:8
40     0, 0, 0,-1, 1, 0, 0, 0,& ! @v case3 i=1:4,j=5:8
41     0, 0, 1, 0, 1, 0, 0, 0 & ! @v case4 i=1:4,j=5:8
42     /),(/8,5,2/)) ! 0:4 → 5
43 integer, allocatable :: ij_cv(:,,:),ij_u(:,,:),ij_v(:,,:),ij_u_we(:,,:),ij_v_sn(:,,:), &
44     ij_w(:,,:),ij_e(:,,:),ij_s(:,,:),ij_n(:,,:)
45 integer :: n_ij_cv,n_ij_u,n_ij_v,n_ij_u_we,n_ij_v_sn, &
46     n_ij_w,n_ij_e,n_ij_s,n_ij_n
47 ! → 1d
48 integer, allocatable :: ij_1d2d(:,,:),ivin(:),i_1d(:,,:)
49 real(8), allocatable :: xy_1d2d(:,,:),va_in0(:,,:),vcc_in0(:,,:),vcf_in0(:,,:), &
50     va_in(:),vcc_in(:),vcf_in(:)

```

```

51 integer :: iend_ld, itmp, n_livin, itr_vin, io, n_livin0, itmp1, itmp2
52 real(8) :: va_acc, vcc_acc, vcf_acc
53 ! ← 1d
54 real(8), allocatable :: rseries(:, :, :), r(:, :), rmap(:, :, :), x_rpnt(:), y_rpnt(:), r_rpnt(:),
      d_idw(:)
55 real(8) :: rdt, rain_tmp
56 integer :: l, rnum, itsp_r, n_rpnt, minl(1)
57 character(len=100) :: rname
58 !
59 integer :: nsci, nscj
60 integer, allocatable :: iff_x(:, :), iff_y(:, :)
61 real(8), allocatable :: mvals(:)
62 character :: cline*1000
63 !
64 character(len=100) :: fname, ctmp, cfmt, out_dir
65 !
66 open(1, file='DF_input.txt')
67 read(1, *) ! grid
68 read(1, *) iend
69 read(1, *) jend
70 read(1, *) dx
71 read(1, *) dy
72 !
73 allocate(z(iend, jend), zo(iend, jend), zini(iend, jend), zs(iend, jend), h(iend, jend), ho(iend, jend),
      d(iend, jend), &
74          hs(iend, jend), hso(iend, jend), coh(iend, jend), sf(iend, jend), hmax(iend, jend), hsmx(
      iend, jend), &
75          cc(iend, jend), cco(iend, jend), cf(iend, jend), cfo(iend, jend), e(iend, jend), tant_e(iend,
      jend), &
76          grad_z(iend, jend), grad_w(iend, jend), rho(iend, jend), rhom(iend, jend), fdir(iend, jend))
77 allocate(u(iend, jend), uo(iend, jend), v(iend, jend), vo(iend, jend), ua(iend, jend), va(iend, jend),
      uv(iend, jend), &
78          uh(iend, jend), vh(iend, jend), qx(iend, jend), qy(iend, jend), qsx(iend, jend), qsy(iend,
      jend), &
79          qccx(iend, jend), qccy(iend, jend), qcfx(iend, jend), qcfy(iend, jend), &
80          q_a(iend, jend, 2), qcc_a(iend, jend, 2), qcf_a(iend, jend, 2))
81 allocate(qin0(0:qnum), cin0(0:qnum))
82 allocate(iacc(iend, jend), ibasin(iend, jend), idir(iend, jend), ifld(iend, jend))
83 !
84 read(1, *)
85 read(1, *) dt
86 read(1, *) dout_time
87 read(1, *) fout_time
88 read(1, *)
89 read(1, *) rname
90 read(1, *) rnum
91 read(1, *) rdt
92 read(1, *)
93 read(1, *) qnum
94 read(1, *) qdt
95 read(1, *)
96 read(1, *) fname ! elevation
97 call r_fasc(10, fname, z(:, :), iend, jend, xllcorner, yllcorner, cellsize)
98 !
99 read(1, *) fname ! targetArea

```

```

100 call r_iasc(10,fname,ibasin(:,,:),iend,jend,xllcorner,yllcorner,cellsize)
101 !
102 ifld(:,:)=0
103 !fname='input/fld.asc'
104 !call r_iasc(10,fname,ifld(:,,:),iend,jend,xllcorner,yllcorner,cellsize)
105 !open(11,file='ifld-in.txt')
106 !read(11,*)nifld
107 !do k=1,nifld
108 ! read(11,*)i,j
109 ! ifld(i,j)=1
110 !end do
111 !
112 read(1,*)fname ! flowDir
113 call r_iasc(10,fname,idir(:,,:),iend,jend,xllcorner,yllcorner,cellsize)
114 fdir(:,:)=idir(:,:)*45*deg2rad
115 !
116 read(1,*)fname ! soil depth(m)
117 call r_fasc(10,fname,d(:,,:),iend,jend,xllcorner,yllcorner,cellsize)
118 zs(:,:)=z(:,:)-d(:,:)
119 zini(:,:)=z(:,:)
120 !
121 read(1,*)fname ! initial hs(m)
122 call r_fasc(10,fname,hs,iend,jend,xllcorner,yllcorner,cellsize)
123 hso(:,:)=hs(:,:)
124 !
125 read(1,*)
126 read(1,*)dm
127 read(1,*)csta
128 read(1,*)pf
129 read(1,*)rhow
130 read(1,*)sig
131 read(1,*)tanp ! deg
132 tanp=tan(tanp*deg2rad) ! deg → rad
133 read(1,*)coh_tmp
134 read(1,*)ks ! (mm/h)
135 ks=ks/100.0d0 ! (mm/h) → (m/s)
136 read(1,*)rn
137 read(1,*)hlim
138 !
139 cstad=csta
140 pc=1.d0-pf
141 end_time=dbl(qnum-1)*qdt
142 itsp_max=nint(end_time/dt)
143 itsp_dout=nint(dout_time/dt)
144 itsp_fout=nint(fout_time/dt)
145 itsp_q=nint(qdt/dt)
146 itsp_r=nint(rdt/dt)
147 out_dir='output_DF/'
148 write(*,*)trim(adjustl(out_dir))
149 write(*,*)''
150 !
151 read(1,*)
152 read(1,*)nsci
153 allocate(iffx(nsci,3))
154 write(cline,'(a10)')'time'

```

```

155 do i=1,nsci
156   read(1,*) iffxx(i,1), iffxx(i,2), iffxx(i,3) ! i,j1,j2
157   write(ctmp, '(a,i0,a,i0,a,i0,a)') i(' ', iffxx(i,1), ')&j(' ', iffxx(i,2), ':', iffxx(i,3), '),',
158   ctmp=adjustr(ctmp)
159   cline=trim(cline)//ctmp(56:100)
160 end do
161 !
162 read(1,*) nscj
163 allocate(iffy(nscj,3))
164 do i=1,nscj
165   read(1,*) iffy(i,1), iffy(i,2), iffy(i,3)
166   write(ctmp, '(a,i0,a,i0,a,i0,a)') i(' ', iffy(i,2), ':', iffy(i,3), ')&j(' ', iffy(i,1), '),',
167   ctmp=adjustr(ctmp)
168   cline=trim(cline)//ctmp(56:100)
169 end do
170 open(30, file='flux_sed.txt')
171 write(30, '(a)') cline
172 !
173 allocate(mvals((nsci+nscj)*3))
174 !
175 ! —> set cv & flux
176 open(1, file='input/controlVolume_centerPoint_inDF.txt')
177 read(1,*) n_ij_cv
178 allocate( ij_cv(n_ij_cv,2))
179 read(1,*)
180 do k=1,n_ij_cv
181   read(1,*) x,y, ij_cv(k,1:2)
182 end do
183 close(1)
184 !
185 open(1, file='input/fluxPoint_x_inDF.txt')
186 read(1,*) n_ij_u
187 allocate( ij_u(n_ij_u,2))
188 read(1,*)
189 do k=1,n_ij_u
190   read(1,*) x,y, ij_u(k,1:2)
191 end do
192 close(1)
193 !
194 open(1, file='input/fluxPoint_y_inDF.txt')
195 read(1,*) n_ij_v
196 allocate( ij_v(n_ij_v,2))
197 read(1,*)
198 do k=1,n_ij_v
199   read(1,*) x,y, ij_v(k,1:2)
200 end do
201 close(1)
202 !
203 open(1, file='input/fluxPoint_xBoundary_inDF.txt')
204 read(1,*) n_ij_u_we
205 allocate( ij_u_we(n_ij_u_we,2))
206 read(1,*)
207 do k=1,n_ij_u_we
208   read(1,*) x,y, ij_u_we(k,1:2)
209 end do

```



```

210 close(1)
211 !
212 open(1, file='input/fluxPoint_yBoudary_inDF.txt')
213 read(1,*) n_ij_v_sn
214 allocate( ij_v_sn(n_ij_v_sn,2))
215 read(1,*)
216 do k=1,n_ij_v_sn
217   read(1,*)x,y, ij_v_sn(k,:)
218 end do
219 close(1)
220 !
221 open(1, file='input/fluxPoint_wBoudary_inDF.txt')
222 read(1,*) n_ij_w
223 allocate( ij_w(n_ij_w,2))
224 read(1,*)
225 do k=1,n_ij_w
226   read(1,*)x,y, ij_w(k,:)
227 end do
228 close(1)
229 !
230 open(1, file='input/fluxPoint_eBoudary_inDF.txt')
231 read(1,*) n_ij_e
232 allocate( ij_e(n_ij_e,2))
233 read(1,*)
234 do k=1,n_ij_e
235   read(1,*)x,y, ij_e(k,:)
236 end do
237 close(1)
238 !
239 open(1, file='input/fluxPoint_sBoudary_inDF.txt')
240 read(1,*) n_ij_s
241 allocate( ij_s(n_ij_s,2))
242 read(1,*)
243 do k=1,n_ij_s
244   read(1,*)x,y, ij_s(k,:)
245 end do
246 close(1)
247 !
248 open(1, file='input/fluxPoint_nBoudary_inDF.txt')
249 read(1,*) n_ij_n
250 allocate( ij_n(n_ij_n,2))
251 read(1,*)
252 do k=1,n_ij_n
253   read(1,*)x,y, ij_n(k,:)
254 end do
255 close(1)
256 ! <— set cv & flux
257 !
258 !open(10, file='input/bound.dat')
259 !do i=0,qnum ! i=0: initial
260 ! read(10,*)qin0(i),cin0(i)
261 !end do
262 !qin0(qnum+1)=qin0(qnum)
263 !cin0(qnum+1)=cin0(qnum)
264 !close(10)

```

```

265 !
266 ! —> set vol from Id
267 open(10, file='input/streamConfiguration_inRR.txt')
268 read(10,*) iend_1d
269 allocate(ij_1d2d(iend_1d,2,1), xy_1d2d(iend_1d,2))
270 read(10,*)
271 do i=1,iend_1d
272   read(10,*) itmp, itmp, itmp, itmp, itmp, &
273         tmp, tmp, tmp, tmp, tmp, tmp, itmp, &
274         xy_1d2d(i,1), xy_1d2d(i,2), ij_1d2d(i,1,1), ij_1d2d(i,2,1)
275 end do
276 close(10)
277 deallocate(ij_1d2d)
278 !
279 open(10, file='input/streamFloodplainConnection.txt')
280 read(10,*) itr_vin
281 read(10,*)
282 itmp=0
283 io=0
284 do while(io==0)
285   read(10,*, iostat=io)
286   itmp=itmp+1
287 end do
288 close(10)
289 itmp=itmp-1
290 n_ivin=itmp
291 if(mod(n_ivin, itr_vin)/=0) then
292   write(*,*) 'n%itr/=0_<<<STOP>>>'
293   stop
294 else
295   n_ivin=n_ivin/itr_vin
296   n_ivin0=n_ivin
297 end if
298 !
299 allocate(ij_1d2d(n_ivin,2, itr_vin), i_1d(n_ivin, itr_vin))
300 open(10, file='input/streamFloodplainConnection.txt')
301 read(10,*)
302 read(10,*)
303 do i=1, n_ivin
304   do j=1, itr_vin
305     read(10,*) tmp, tmp, ij_1d2d(i,1,j), ij_1d2d(i,2,j), i_1d(i,j)
306   end do
307 end do
308 close(10)
309 !
310 allocate(va_in(iend_1d))
311 open(10, file='./input/flowVolume_ws_DRtoDF.txt')
312 do i=1, qnum
313   read(10, '(a10,i10,*(E20.10e3)') ctmp, k, va_in(:)
314 end do
315 close(10)
316 !
317 itmp=0
318 do i=1, n_ivin
319   if(0.d0 < va_in(i_1d(i,1))) then

```

```

320     itmp=itmp+1
321 end if
322 end do
323 n_ivin=itmp
324 !
325 deallocate(ij_1d2d,i_1d)
326 allocate(ivin(n_ivin))
327 allocate(ij_1d2d(n_ivin,2,itr_vin),i_1d(n_ivin,itr_vin))
328 !
329 open(10,file='input/streamFloodplainConnection.txt')
330 read(10,*)
331 read(10,*)
332 k=0
333 do i=1,n_ivin0
334     j=1
335     read(10,*)tmp,tmp,itmp,itmp,itmp1,itmp2
336     backspace(10)
337     if(0.d0 < va_in(itmp1))then
338         k=k+1
339         do j=1,itr_vin
340             read(10,*)tmp,tmp,ij_1d2d(k,1,j),ij_1d2d(k,2,j),i_1d(k,j)
341         end do
342         if(i_1d(k,1)==i_1d(k,itr_vin))then
343             ivin(k)=i_1d(k,1)
344         else
345             write(*,*)'n_ivin?_<<<STOP>>>'
346             stop
347         end if
348     else
349         do j=1,itr_vin
350             read(10,*)
351         end do
352     end if
353 end do
354 deallocate(va_in)
355 !
356 allocate(va_in0(0:qnum,iend_1d),vcc_in0(0:qnum,iend_1d),vcf_in0(0:qnum,iend_1d), &
357         va_in(iend_1d),vcc_in(iend_1d),vcf_in(iend_1d))
358 va_in0=0.d0
359 vcc_in0=0.d0
360 vcf_in0=0.d0
361 va_in=0.d0
362 vcc_in=0.d0
363 vcf_in=0.d0
364 open(10,file='./input/flowVolume_ws_DRtoDF.txt')
365 do i=0,qnum-1 ! i=0: initial
366     read(10,'(a10,i10,*(E20.10e3))')ctmp,k,va_in0(i,:)
367     if(i==0)then
368         va_in(:)=va_in0(i,:)
369     else
370         va_in0(i,:)=va_in0(i:)-va_in(:)
371     end if
372     va_in(:)=va_in(:)+va_in0(i,:)
373 end do
374 close(10)

```

```

375 va_in0(qnum,:)=va_in0(qnum-1,:)
376 va_in0(:,:)=va_in0(:,:)/(cellsize*cellsize*qdt)
377 !
378 open(10,file='./input/flowVolume_cs_DRtoDF.txt')
379 do i=0,qnum-1 ! i=0: initial
380   read(10,'(a10,i10,(E20.10e3))')ctmp,k,vcc_in0(i,:)
381   if(i==0)then
382     vcc_in(:)=vcc_in0(i,:)
383   else
384     vcc_in0(i,:)=vcc_in0(i,)-vcc_in(:)
385   end if
386   vcc_in(:)=vcc_in(:)+vcc_in0(i,:)
387 end do
388 close(10)
389 vcc_in0(qnum,:)=vcc_in0(qnum-1,:)
390 vcc_in0(:,:)=vcc_in0(:,:)/(cellsize*cellsize*qdt)
391 !
392 open(10,file='./input/flowVolume_fs_DRtoDF.txt')
393 do i=0,qnum-1 ! i=0: initial
394   read(10,'(a10,i10,(E20.10e3))')ctmp,k,vcf_in0(i,:)
395   if(i==0)then
396     vcf_in(:)=vcf_in0(i,:)
397   else
398     vcf_in0(i,:)=vcf_in0(i,)-vcf_in(:)
399   end if
400   vcf_in(:)=vcf_in(:)+vcf_in0(i,:)
401 end do
402 close(10)
403 vcf_in0(qnum,:)=vcf_in0(qnum-1,:)
404 vcf_in0(:,:)=vcf_in0(:,:)/(cellsize*cellsize*qdt)
405 !
406 open(10,file='va.txt')
407 write(10,'((i10),(a10),(i20))')n_ivin*itr_vin,'i_1D',((i_1d(j,k),k=1,itr_vin),j=1,n_ivin)
408 write(10,'((i10),(a10),(i20))')n_ivin*itr_vin,'i_2D',((ij_1d2d(j,1,k),k=1,itr_vin),j=1,
n_ivin)
409 write(10,'((i10),(a10),(i20))')n_ivin*itr_vin,'j_2D',((ij_1d2d(j,2,k),k=1,itr_vin),j=1,
n_ivin)
410 do i=1,qnum
411   write(10,'((a10),(i10),(E20.10e3))')time(s)=',i*int(qdt),((va_in0(i,i_1d(j,k)))/dble(
itr_vin),k=1,itr_vin),j=1,n_ivin)
412 end do
413 close(10)
414 !
415 open(10,file='vcc.txt')
416 write(10,'((i10),(a10),(i20))')n_ivin*itr_vin,'i_1D',((i_1d(j,k),k=1,itr_vin),j=1,n_ivin)
417 write(10,'((i10),(a10),(i20))')n_ivin*itr_vin,'i_2D',((ij_1d2d(j,1,k),k=1,itr_vin),j=1,
n_ivin)
418 write(10,'((i10),(a10),(i20))')n_ivin*itr_vin,'j_2D',((ij_1d2d(j,2,k),k=1,itr_vin),j=1,
n_ivin)
419 do i=1,qnum
420   write(10,'((a10),(i10),(E20.10e3))')time(s)=',i*int(qdt),((vcc_in0(i,i_1d(j,k)))/dble(
itr_vin),k=1,itr_vin),j=1,n_ivin)
421 end do
422 close(10)
423 !

```

```

424 open(10, file='vcf.txt')
425 write(10, '((i10), (a10), *(i20))' n_ivin*itr_vin, 'i_1D', ((i_1d(j, k), k=1, itr_vin), j=1, n_ivin)
426 write(10, '((i10), (a10), *(i20))' n_ivin*itr_vin, 'i_2D', ((ij_1d2d(j, 1, k), k=1, itr_vin), j=1,
n_ivin)
427 write(10, '((i10), (a10), *(i20))' n_ivin*itr_vin, 'j_2D', ((ij_1d2d(j, 2, k), k=1, itr_vin), j=1,
n_ivin)
428 do i=1, qnum
429   write(10, '((a10), (i10), *(E20.10e3))' time(s)=, i*int(qdt), ((vcf_in0(i, i_1d(j, k)))/dble(
itr_vin), k=1, itr_vin), j=1, n_ivin)
430 end do
431 close(10)
432 !
433 open(10, file='connection_mod.txt')
434 write(10, *) n_ivin*itr_vin
435 do i=1, n_ivin
436   do j=1, itr_vin
437     x=xlcorner+cellsize*(0.5+ij_1d2d(i, 1, j)-1)
438     y=yllcorner+cellsize*(0.5+ij_1d2d(i, 2, j)-1)
439     write(10, '(2(f15.3, ', ', '), 5(i5, ', ', '), ', x, y, (i-1)*itr_vin+j, ij_1d2d(i, 1, j), ij_1d2d(i, 2, j
), i_1d(i, j), ivin(i)
440   end do
441 end do
442 close(10)
443 ! <— set vol from 1d
444 !
445 ! —> reconfiguration of input vol
446 deallocate(ij_1d2d)
447 open(10, file='connection_mod.txt')
448 read(10, *) n_ivin
449 allocate(ij_1d2d(n_ivin, 2, 1))
450 do i=1, n_ivin
451   read(10, *) x, y, itmp, ij_1d2d(i, 1, 1), ij_1d2d(i, 2, 1)
452 end do
453 !
454 deallocate(va_in0, vcc_in0, vcf_in0, va_in, vcc_in, vcf_in)
455 allocate(va_in0(0:qnum, n_ivin), vcc_in0(0:qnum, n_ivin), vcf_in0(0:qnum, n_ivin), &
va_in(n_ivin), vcc_in(n_ivin), vcf_in(n_ivin))
457 va_in0=0.d0
458 vcc_in0=0.d0
459 vcf_in0=0.d0
460 va_in=0.d0
461 vcc_in=0.d0
462 vcf_in=0.d0
463 open(10, file='va.txt')
464 read(10, *)
465 read(10, *)
466 read(10, *)
467 do i=1, qnum ! i=0: initial
468   read(10, '((a10), (i10), *(E20.10e3))' ctmp, k, va_in0(i, 1: n_ivin)
469 end do
470 close(10)
471 !
472 open(10, file='vcc.txt')
473 read(10, *)
474 read(10, *)

```

```

475 read(10,*)
476 do i=1,qnum ! i=0: initial
477   read(10,'((a10),(i10),(E20.10e3)')ctmp,k,vcc_in0(i,1:n_ivin)
478 end do
479 close(10)
480 !
481 open(10,file='vcf.txt')
482 read(10,*)
483 read(10,*)
484 read(10,*)
485 do i=1,qnum ! i=0: initial
486   read(10,'((a10),(i10),(E20.10e3)')ctmp,k,vcf_in0(i,1:n_ivin)
487 end do
488 close(10)
489 ! <— reconfiguration of input vol
490 !
491 ! —> set rain
492 open(11,file=rname)
493 read(11,*)n_rpnt
494 allocate(rseries(iend,jend,0:rnum),rmap(iend,jend,0:n_rpnt),r(iend,jend), &
495          x_rpnt(n_rpnt),y_rpnt(n_rpnt),r_rpnt(n_rpnt),d_idw(n_rpnt))
496 read(11,*)ctmp,x_rpnt(1:n_rpnt)
497 read(11,*)ctmp,y_rpnt(1:n_rpnt)
498 rseries(:, :, 0:rnum)=0.0d0 ! from 0
499 rmap(:, :, :)=0
500 do i=1,iend
501   do j=1,jend
502     if(ibasin(i,j)/=0)then
503       x=xllcorner+dx*(0.5+i-1)
504       y=yllcorner+dx*(0.5+j-1)
505       d_idw(1:n_rpnt)=((x-x_rpnt(1:n_rpnt))**2.d0+(y-y_rpnt(1:n_rpnt))**2.d0)**0.5d0
506       minl=minloc(d_idw)
507       !write(*,*)i,j,minl
508       rmap(i,j,1:n_rpnt)=0.d0
509       rmap(i,j,minl(1))=1.d0 !d_idw(1:n_rpnt)/sum(d_idw(:))
510       rmap(i,j,0)=minl(1)
511     end if
512   end do
513 end do
514 fname=trim(adjustl(out_dir))// 'rain_map.asc'
515 cfmt='(f10.3) '
516 call w_fasc_fmt(20,fname,rmap(:, :, 0),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.d0)
517
518 do k=0,rnum-1
519   rseries(:, :, k)=0.0d0
520   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
521   read(11,*)ctmp,r_rpnt(1:n_rpnt)
522   do i=1,iend
523     do j=1,jend
524       if(ibasin(i,j)/=0)then
525         rain_tmp=sum(r_rpnt(1:n_rpnt)*rmap(i,j,1:n_rpnt))
526         !write(*,'(2i4,10f7.3)')i,j,rain_tmp,r_rpnt(:),rmap(i,j,:)
527         rseries(i,j,k)=rain_tmp/rdt/1000.0d0
528       end if
529     end do

```

```

530  end do
531  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
532  !rseries(1:iend,1:jend,k)=rain_tmp/rdt/1000.0d0
533  !where(ibasin(1:iend,1:jend)==0) rseries(1:iend,1:jend,k)=0.0d0
534  !write(fname,'(i10.10)')nint(k*rdt)
535  !fname='output_rain/rain_'//trim(adjustl(fname))//'.asc'
536  !write(*,*)trim(adjustl(fname)), ' mm/h'
537  !call w_fasc(20,fname,cfmt,rseries(:, :, k)*3600.*1000.0d0,iend,jend,xllcorner,yllcorner,dx
    , -9999.0d0)
538  !open(10,file=fname)
539  !do i=1,6
540  !  read(10,*)
541  !end do
542  !do j=jend,1,-1
543  !  read(10,*)rseries(1:iend,j,k)
544  !  rseries(1:iend,j,k)=rseries(1:iend,j,k)/rdt/1000.0d0
545  !end do
546  end do
547
548  fname=trim(adjustl(out_dir))//'rain_sum.asc'
549  write(*,*)trim(adjustl(fname)), ' _mm'
550  r=0.d0
551  do k=0,rnum-1
552  r(:, :)=r(:, :)+(rseries(:, :, k))*rdt*1000.
553  end do
554  !write(*,*)'r_ave=',sum(r(:, :))*dx*dx/basin_area
555  call w_fasc_fmt(20,fname,r(:, :),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.0d0)
556  r=0.d0
557  close(11)
558  rseries(:, :, rnum)=rseries(:, :, rnum-1)
559  ! <— set rain
560  !
561  ! —> initial conditions
562  h(:, :)=0.d0
563  u(:, :)=0.d0
564  v(:, :)=0.d0
565  uh(:, :)=0.d0
566  vh(:, :)=0.d0
567  ho(:, :)=h(:, :)
568  uo(:, :)=u(:, :)
569  vo(:, :)=v(:, :)
570  zo(:, :)=z(:, :)
571  cc(:, :)=0.d0
572  cf(:, :)=0.d0
573  cco(:, :)=cc(:, :)
574  cfo(:, :)=cf(:, :)
575  e(:, :)=0.d0
576  q_a(:, :, 1:2)=0.d0
577  qcc_a(:, :, 1:2)=0.d0
578  qcf_a(:, :, 1:2)=0.d0
579  !hs(:, :)=0.d0
580  coh(:, :)=coh_tmp
581  sf(:, :)=0.d0
582  hmax=0.d0
583  hsmax=0.d0

```

```

584 umax=0.d0
585 vmax=0.d0
586 i_umax=0
587 j_umax=0
588 h_umax=0.d0
589 i_vmax=0
590 j_vmax=0
591 h_vmax=0.d0
592 va_acc=0.d0
593 vcc_acc=0.d0
594 vcf_acc=0.d0
595 q_out=0.d0
596 qcc_out=0.d0
597 qcf_out=0.d0
598 ! <— initial conditions
599 !
600 ! —> cal start
601 dt0=dt
602 dtmin=dt
603 time=0.d0
604 time0=0.d0
605 do itsp=1,itsp_max
606   time=time0+dt
607   k=itsp/itsp_q
608   l=itsp/itsp_r
609   qin=qin0(k)+(qin0(k+1)-qin0(k))*(itsp-k*itsp_q)/dbble(itsp_q)
610   cin=cin0(k)+(cin0(k+1)-cin0(k))*(itsp-k*itsp_q)/dbble(itsp_q)
611   r(:,:)=rseries(:,:,l+1)
612   va_in(:)=va_in0(k,:)+(va_in0(k+1,:)-va_in0(k,:))*(itsp-k*itsp_q)/dbble(itsp_q)
613   vcc_in(:)=vcc_in0(k,:)+(vcc_in0(k+1,:)-vcc_in0(k,:))*(itsp-k*itsp_q)/dbble(itsp_q)
614   vcf_in(:)=vcf_in0(k,:)+(vcf_in0(k+1,:)-vcf_in0(k,:))*(itsp-k*itsp_q)/dbble(itsp_q)
615   !
616   time1=time0
617   itr=1
618   flxd=0.d0
619   !
620   call cal_qs(n_ij_u , ij_u , iend , jend , -1 , 0 , dx , ks , zs , hs , qsx , usmax)
621   call cal_qs(n_ij_v , ij_v , iend , jend , 0 , -1 , dx , ks , zs , hs , qsy , vsmax)
622   ! flxd=max(usmax*dt , vsmax*dt , ks*dt)
623   ! flxd=max(flxd , maxval(abs(qsx))*dt , maxval(abs(qsy))*dt)
624   !
625   call cal_rho(n_ij_cv , ij_cv , iend , jend , sig , rhov , cc , cf , rho , rhom) ! i=i to iend
626   !u
627   call cal_u(n_ij_u , ij_u , iend , jend , dtlim , dx , g , zo , h , ho , tant_e , hlim , rn , u , uo , uh , &
628             rho , sig , dm , csta , tanp , cco , ijofs_uv(1:8,0:4,1) , vo , umax , i_umax , j_umax , h_umax)
629   !v
630   call cal_u(n_ij_v , ij_v , iend , jend , dtlim , dx , g , zo , h , ho , tant_e , hlim , rn , v , vo , vh , &
631             rho , sig , dm , csta , tanp , cco , ijofs_uv(1:8,0:4,2) , uo , vmax , i_vmax , j_vmax , h_vmax)
632   !
633   ! flxd=max(flxd , umax*h_umax*dt , vmax*h_vmax*dt)
634   flxd=max(umax*dt , vmax*dt)
635   ! tmp=u2*maxval(va_in(:))*(2.d0*g*maxval(va_in(:)))*0.5d0
636   tmp=(g*maxval(va_in(:)))*0.5d0
637   flxd=max(flxd , tmp*dt)
638   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

```



```

639  if ((dx*cfl) < flxd)then
640      itr=min(ceiling(flxd/(dx*cfl)),nint(dt0/dtlim))
641      if(itr<=0)then
642          write(*,*)'itr',itr,flxd,dx*cfl,nint(dt0/dtlim)
643          write(*,*)tmp,h_umax,h_vmax,maxval(va_in(:))
644          read(*,*)
645      end if
646      dt=dt0/dble(itr)
647      !write(*,'(a,f10.7)',advance='yes')'dt=',dt
648      if(dt<dtmin)dtmin=dt
649  end if
650  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
651  !
652  do while (itr>0)
653      !
654      !d(:,:)=z(:,:)-zs(:,:)
655      !call cal_hs(n_ij_cv,ij_cv,iend,jend,dt,dx,hs,r(:,:),qsx,qsy,csta)
656      !where(d(:,:) < hs(:,:))
657      ! ho(:,:)=ho(:,:)+(hs(:,:)-d(:,:))*(1.d0-csta)
658      ! hs(:,:)=d(:,:)
659      !elsewhere
660      !end where
661      !
662      !call cal_stability(n_ij_cv,ij_cv,iend,jend,&
663      !                    grad_w,h,hs,&
664      !                    (1.d0-csta),rhow,sig,d,tanp,coh,g,sf)
665      !where(0.d0 < e(:,:) .and. hs(:,:) < d(:,:))
666      ! e(:,:)=0.d0
667      !elsewhere
668      !end where
669      !
670      call cal_h(n_ij_cv,ij_cv,iend,jend,dt,dx,h,ho,qx,qy,e,csta,cstad)
671      call cal_cc(n_ij_cv,ij_cv,iend,jend,dt,dx,hlim,h,ho,cc,cco,qccx,qccy,e,pc)
672      call cal_cf(n_ij_cv,ij_cv,iend,jend,dt,dx,hlim,h,ho,cf,cfo,cc,cco,qcfx,qcfy,e,pf,cstad)
673      call cal_z(n_ij_cv,ij_cv,iend,jend,dt,csta,z,zs,e,tant_e)
674      !
675      ! —> from 1d
676      do k=1,n_ivin
677          i=ij_1d2d(k,1,1)
678          j=ij_1d2d(k,2,1)
679          ho(i,j)=h(i,j)
680          cco(i,j)=cc(i,j)
681          cfo(i,j)=cf(i,j)
682          h(i,j)=ho(i,j)+va_in(k)*dt
683          if(0.d0<h(i,j))then
684              cc(i,j)=(cco(i,j) *ho(i,j) + vcc_in(k)*dt) / h(i,j)
685              cf(i,j)=((1.d0-cco(i,j))*ho(i,j)*cfo(i,j) + vcf_in(k)*dt) / ((1.d0-cc(i,j))*h(i,j)
686          ))
687          else
688              cc(i,j)=0.d0
689              cf(i,j)=0.d0
690          end if
691          va_acc = va_acc+ va_in(k)*dt*cellsize*cellsize
692          vcc_acc=vcc_acc+vcc_in(k)*dt*cellsize*cellsize
693          vcf_acc=vcf_acc+vcf_in(k)*dt*cellsize*cellsize

```

```

693 end do
694 ! ← from 1d
695 !
696 call cal_qs(n_ij_u , ij_u , iend , jend , -1, 0, dx, ks , zs , hs , qsx , usmax)
697 call cal_qs(n_ij_v , ij_v , iend , jend , 0, -1, dx, ks , zs , hs , qsy , vsmx)
698 !u
699 call cal_u(n_ij_u , ij_u , iend , jend , dt, dx, g, zo, h, ho, tant_e , hlim , rn , u, uo, uh, &
700          rho , sig , dm, csta , tanp , cco , ijofs_uv (1:8, 0:4, 1) , vo, umax , i_umax , j_umax , h_umax)
701 !v
702 call cal_u(n_ij_v , ij_v , iend , jend , dt, dx, g, zo, h, ho, tant_e , hlim , rn , v, vo, vh, &
703          rho , sig , dm, csta , tanp , cco , ijofs_uv (1:8, 0:4, 2) , uo, vmax , i_vmax , j_vmax , h_vmax)
704 !
705 ua(1:iend-1,:)=u(1:iend-1,:)+u(2:iend,:)*0.5d0
706 ua(iend,:)=u(iend,:)
707 va(:,1:jend-1)=(v(:,1:jend-1)+v(:,2:jend))*0.5d0
708 va(:,jend)=v(:,jend)
709 uv=(ua*ua+va*va)**0.5d0
710 !
711 call cal_q(n_ij_u , ij_u , iend , jend , -1, 0, h, u, qx, cc, cf, qccx, qcfx, uh)
712 call cal_q(n_ij_v , ij_v , iend , jend , 0, -1, h, v, qy, cc, cf, qccy, qcfy, vh)
713 !
714 call cal_rho(n_ij_cv , ij_cv , iend , jend , sig , rhow, cc, cf, rho, rhom) ! i=i to iend
715 !
716 call cal_grad(n_ij_cv , ij_cv , iend , jend , dx, z, grad_z)
717 call cal_grad(n_ij_cv , ij_cv , iend , jend , dx, zs+hs+h, grad_w)
718 !
719 call cal_e(n_ij_cv , ij_cv , iend , jend , dt, dx, z, zs, sig, rho, csta, tanp, cc, cf, e, h, hlim, &
720          ua, va, qx, qy, qccx, qccy, qcfx, qcfy, ifld, tant_e, uv) ! output: i=1 to iend-1
721 !
722 uo(:,:)=u(:,:)
723 vo(:,:)=v(:,:)
724 ho(:,:)=h(:,:)
725 zo(:,:)=z(:,:)
726 cco(:,:)=cc(:,:)
727 cfo(:,:)=cf(:,:)
728 !
729 q_a(:,:,1)=q_a(:,:,1)+qx(:,:)*dt
730 q_a(:,:,2)=q_a(:,:,2)+qy(:,:)*dt
731 qcc_a(:,:,1)=qcc_a(:,:,1)+qccx(:,:)*dt
732 qcc_a(:,:,2)=qcc_a(:,:,2)+qccy(:,:)*dt
733 qcf_a(:,:,1)=qcf_a(:,:,1)+qcfx(:,:)*dt
734 qcf_a(:,:,2)=qcf_a(:,:,2)+qcfy(:,:)*dt
735 !
736 do k=1, n_ij_w
737   q_out=q_out-qx(ij_w(k,1), ij_w(k,2))*dt*dx
738   qcc_out=qcc_out-qccx(ij_w(k,1), ij_w(k,2))*dt*dx
739   qcf_out=qcf_out-qcfx(ij_w(k,1), ij_w(k,2))*dt*dx
740 end do
741 do k=1, n_ij_e
742   q_out=q_out+qx(ij_e(k,1), ij_e(k,2))*dt*dx
743   qcc_out=qcc_out+qccx(ij_e(k,1), ij_e(k,2))*dt*dx
744   qcf_out=qcf_out+qcfx(ij_e(k,1), ij_e(k,2))*dt*dx
745 end do
746 do k=1, n_ij_s
747   q_out=q_out-qy(ij_s(k,1), ij_s(k,2))*dt*dx

```

```

748     qcc_out=qcc_out-qccy( ij_s(k,1), ij_s(k,2))*dt*dx
749     qcf_out=qcf_out-qcfy( ij_s(k,1), ij_s(k,2))*dt*dx
750 end do
751 do k=1,n_ij_n
752     q_out =q_out +qy( ij_n(k,1), ij_n(k,2))*dt*dx
753     qcc_out=qcc_out+qccy( ij_n(k,1), ij_n(k,2))*dt*dx
754     qcf_out=qcf_out+qcfy( ij_n(k,1), ij_n(k,2))*dt*dx
755 end do
756 !
757 where(hmax(:, : )<h( : , : ))hmax( : , : )=h( : , : )
758 where(hsmax(:, : )<hs( : , : ))hsmax( : , : )=hs( : , : )
759 !
760     time1=time1+dt
761     itr=itr-1
762     !write(*, '(a,3f10.5,i5)')'in do while1, time,time0,time1',time,time0,time1,itr
763 end do ! do while
764 !write(*, '(a,3f10.5)')'af do while1, time,time0,time1',time,time0,time1
765 !
766     time0=time
767     dt=dt0
768     ! —> output
769 if (mod(itsp, itsp_dout) ==0) then
770     ! —> to disp
771     write(*, '(a,f10.3)')      'time(day)_=' ,time/86400.d0
772     write(*, '(a,f10.3)')      'time(hour)=' ,time/3600.d0
773     write(*, '(a,i10,2(a,f7.3)')'time(sec)_=' ,nint(time), ',_rain(mm/h)=' ,maxval(r
( : : )*1000*3600.),&
774     ',_va_in(m/s)=' ,maxval(va_in(:)/ dble(itr_vin))
775     write(*, '(2(a,f10.3),a,2i5)')'umax=' ,umax, ',_h@umax=' ,h_umax, ',_i,j=' ,i_umax,j_umax
776     write(*, '(2(a,f10.3),a,2i5)')'vmax=' ,vmax, ',_h@vmax=' ,h_vmax, ',_i,j=' ,i_vmax,j_vmax
777     write(*, '(2(a,f10.5)')'dt_min=' ,dtmin, ',_dt_lim=' ,dtlim
778     write(*, '(a,3e20.5e3)')'va_in,_vcc_in,_vcf_in', va_acc, vcc_acc, vcf_acc
779     write(*, '(a,3e20.5e3)')'vasum,_vccsum,_vcfsum', q_out+sum(h)*cellsize*cellsize,&
780     qcc_out+sum(h*cc+(z-zini)*csta)*cellsize*cellsize,&
781     qcf_out+sum(h*(1.d0-cc)*cf)*cellsize*cellsize
782     write(*, '(a,3e20.5e3)')'q_out,qcc_out,qcf_out', q_out, qcc_out, qcf_out
783     ! <— to disp
784     umax=0.d0
785     vmax=0.d0
786     dtmin=dt
787     !
788     ! —> flux at section
789     cline=' '
790     ! —> sec i
791     do i=1,nsci
792         j=(i-1)*3
793         mvals(j+1)=sum(q_a( iff_x(i,1), iff_x(i,2): iff_x(i,3),1))*dx
794         mvals(j+2)=sum(qcc_a( iff_x(i,1), iff_x(i,2): iff_x(i,3),1))*dx
795         mvals(j+3)=sum(qcf_a( iff_x(i,1), iff_x(i,2): iff_x(i,3),1))*dx
796     end do
797     ! <— sec i
798     ! —> sec j
799     do i=1,nscj
800         j=nsci*3+(i-1)*3
801         mvals(j+1)=sum(q_a( iffy(i,2): iffy(i,3), iffy(i,1),2))*dx

```

```

802     mvals(j+2)=sum(qcc_a(iffy(i,2):iffy(i,3),iffy(i,1),2))*dx
803     mvals(j+3)=sum(qcf_a(iffy(i,2):iffy(i,3),iffy(i,1),2))*dx
804 end do
805 ! <— sec i
806 write(30,'(a,i5,20f15.3)')time=', nint(time), mvals
807 ! <— flux at section
808 !
809 ! —> to files
810 if(mod(itstp, itsp_fout) ==0) then
811     ua(1:iend-1,:)=u(1:iend-1,:)+u(2:iend,:)*0.5d0
812     ua(iend,:)=u(iend,:)
813     va(:,1:jend-1)=v(:,1:jend-1)+v(:,2:jend))*0.5d0
814     va(:,jend)=v(:,jend)
815     uv=(ua*ua+va*va)**0.5d0
816     cfmt='*(f10.3) '
817     write(ctmp,'(i10.10)')idnint(time)
818     fname=trim(adjustl(out_dir))//res_h_//trim(adjustl(ctmp))//'.asc'
819     call w_fasc_fmt(20,fname,h(:,,:),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.d0)
820     fname=trim(adjustl(out_dir))//res_hs_//trim(adjustl(ctmp))//'.asc'
821     !call w_fasc_fmt(20,fname,hs(:,,:),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.d0
822 )
823     fname=trim(adjustl(out_dir))//res_hmax.asc'
824     call w_fasc_fmt(20,fname,hmax(:,,:),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.
825 d0)
826     fname=trim(adjustl(out_dir))//res_hsmax.asc'
827     call w_fasc_fmt(20,fname,hsmax(:,,:),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.
828 d0)
829     fname=trim(adjustl(out_dir))//res_sf_//trim(adjustl(ctmp))//'.asc'
830     !call w_fasc_fmt(20,fname,sf(:,,:),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.d0
831 )
832     fname=trim(adjustl(out_dir))//res_cc_//trim(adjustl(ctmp))//'.asc'
833     call w_fasc_fmt(20,fname,cc(:,,:),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.d0)
834     fname=trim(adjustl(out_dir))//res_cf_//trim(adjustl(ctmp))//'.asc'
835     !call w_fasc_fmt(20,fname,cf(:,,:),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.d0
836 )
837     fname=trim(adjustl(out_dir))//res_dz_//trim(adjustl(ctmp))//'.asc'
838     call w_fasc_fmt(20,fname,z(:,:)-zini(:,:),cfmt,iend,jend,xllcorner,yllcorner,cellsize
839 ,-9999.d0)
840     !fname=trim(adjustl(out_dir))//res_e_//trim(adjustl(ctmp))//'.asc'
841     !call w_fasc_fmt(20,fname,e(:,:),'*(f10.3)',iend,jend,xllcorner,yllcorner,cellsize
842 ,-9999.d0)
843     fname=trim(adjustl(out_dir))//res_qx_a_//trim(adjustl(ctmp))//'.asc'
844     !call w_fasc_fmt(20,fname,q_a(:,:),1,cfmt,iend,jend,xllcorner-dx*0.5d0,yllcorner,
845 cellsize,-9999.d0)
846     fname=trim(adjustl(out_dir))//res_qy_a_//trim(adjustl(ctmp))//'.asc'
847     !call w_fasc_fmt(20,fname,q_a(:,:),2,cfmt,iend,jend,xllcorner,yllcorner-dx*0.5d0,
848 cellsize,-9999.d0)
849     fname=trim(adjustl(out_dir))//res_qccx_a_//trim(adjustl(ctmp))//'.asc'
850     !call w_fasc_fmt(20,fname,qcc_a(:,:),1,cfmt,iend,jend,xllcorner-dx*0.5d0,yllcorner,
851 cellsize,-9999.d0)
852     fname=trim(adjustl(out_dir))//res_qccy_a_//trim(adjustl(ctmp))//'.asc'
853     !call w_fasc_fmt(20,fname,qcc_a(:,:),2,cfmt,iend,jend,xllcorner,yllcorner-dx*0.5d0,
854 cellsize,-9999.d0)
855     fname=trim(adjustl(out_dir))//res_qcfx_a_//trim(adjustl(ctmp))//'.asc'

```

```

845     ! call w_fasc_fmt(20,fname,qcf_a(:, :, 1),cfmt,iend,jend,xllcorner-dx*0.5d0,yllcorner ,
      fname=trim(adjustl(out_dir))//'res_qcfy_a_'//trim(adjustl(ctmp))//'.asc'
846     ! call w_fasc_fmt(20,fname,qcf_a(:, :, 2),cfmt,iend,jend,xllcorner-dx*0.5d0,
      ! fname=trim(adjustl(out_dir))//'res_uv_'//trim(adjustl(ctmp))//'.asc'
847     ! call w_fasc_fmt(20,fname,uv(:, :),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.d0
848     ! call w_fasc_fmt(20,fname,uv(:, :),cfmt,iend,jend,xllcorner,yllcorner,cellsize,-9999.d0
849     )
850     ! q_a(:, :, 1:2)=0.d0
851     ! qcc_a(:, :, 1:2)=0.d0
852     ! qcf_a(:, :, 1:2)=0.d0
853     end if
854     ! <— to files
855     end if
856     ! <— output
857     end do
858     ! <— cal start
859     close(30)
860     write(*,*)'---_nomal_end_---'
861     !
862     stop
863     end
864     !
865     !
866     !
867     subroutine cal_u(n_1d, ij_1d, iend, jend, dt, dx, g, zo, h, ho, tan_a, hlim, rn, u, uo, uh, &
      rho, sig, dm, csta, tanp, cco, ijofs, v, umax, i_umax, j_umax, h_umax)
868     ! FX q1  q2  q3  qe
869     !   =>  =>  =>  =>
870     ! CV | z1 | z2 | .... | ze |
871     ! CV | z1 | z2 | .... | ze |
872     use omp_lib
873     implicit none
874     real(8), parameter :: f1d5=1.d0/5.d0, f1d3=1.d0/3.d0, f5d3=5.d0/3.d0, f2d3=2.d0/3.d0, &
875     kf=0.16d0, kd=0.0828d0, e2=0.85d0**2.d0
876     integer :: ni, n_1d, ij_1d(n_1d, 2), i, iend, j, jend, i_umax, j_umax, ijofs(8, 0:4), io(4), jo(4), i2, j2,
      iloc(2), icase
877     real(8) :: va, dt, dx, g, x1, x2, x3, hlim, rn, umax, h_umax, htmp(iend, jend), &
878     zo(iend, jend), h(iend, jend), ho(iend, jend), tan_a(iend, jend), &
879     u(iend, jend), uo(iend, jend), v(iend, jend), h1, hnw, c1, rho1, z1, z2, tan1, &
880     cco(iend, jend), rho(iend, jend), uh(iend, jend), &!, vh(iend, jend)
881     sig, csta, dm, taub, tauy, fb, fb2, tanp, fd, ff, cost, rhom, dhdx
882     !
883     umax=0.d0
884     u(:, :)=0.d0
885     htmp(:, :)=0.d0
886     h_umax=0.d0
887     i_umax=0
888     j_umax=0
889     !$omp parallel do private(ni, i, j, i2, j2, icase, io, jo, &
890     !$omp                                     va, dhdx, h1, hnw, c1, rho1, z1, z2, tan1, x1, x2, x3, &
891     !$omp                                     cost, tauy, fd, ff, fb2, fb, taub, rhom)
892     do ni=1, n_1d
893     i=ij_1d(ni, 1)
894     j=ij_1d(ni, 2)
895     i2=i+ijofs(4, 0)

```

```

896 j2=j+ijofs(8,0)
897 if(ho(i,j)<hlim .and. ho(i2,j2)<hlim)then ! .or. &
898 !h(i,j)<hlim .and. h(i2,j2)<hlim)then
899 u(i,j)=0.d0
900 cycle
901 end if
902 if(0.d0<uo(i,j) .or. uo(i,j)==0.d0 .and. zo(i,j)+ho(i,j) <= zo(i2,j2) +ho(i2,j2) )then
903 h1=ho(i2,j2)
904 hnw=h(i2,j2)
905 c1=cco(i2,j2)
906 rho1=rho(i2,j2)
907 z1=zo(i2,j2)
908 z2=zo(i,j)
909 tan1=tan_a(i,j)
910 else if(uo(i,j)<0.d0 .or. uo(i,j)==0.d0 .and. zo(i2,j2) +ho(i2,j2) < zo(i,j)+ho(i,j))then
911 h1=ho(i,j)
912 hnw=h(i,j)
913 c1=cco(i,j)
914 rho1=rho(i,j)
915 z1=zo(i,j)
916 z2=zo(i2,j2)
917 tan1=tan_a(i2,j2)
918 else
919 write(*,*) '@cal_u',uo(i,j),h1,i,j
920 read(*,*)
921 end if
922 if(h1<hlim .or. z1+h1<=z2)then! .or. hnw<hlim .or. z1+hnw<=z2)then
923 u(i,j)=0.d0
924 cycle
925 end if
926 h1=(ho(i,j)+ho(i2,j2))/2.d0
927 hnw=(h(i,j)+h(i2,j2))/2.d0
928 c1=(cco(i,j)+cco(i2,j2))/2.d0
929 rho1=(rho(i,j)+rho(i2,j2))/2.d0
930 tan1=(tan_a(i,j)+tan_a(i2,j2))/2.d0
931 !if(h1<hlim .or. z1+h1<=z2)then! .or. hnw<hlim .or. z1+hnw<=z2)then
932 ! u(i,j)=0.d0
933 ! cycle
934 !end if
935 io(1:4)=ijofs(1:4,0)
936 jo(1:4)=ijofs(5:8,0)
937 va=v(i+io(1),j+jo(1))+ &
938 v(i+io(2),j+jo(2))+ &
939 v(i+io(3),j+jo(3))+ &
940 v(i+io(4),j+jo(4)) / 4.d0
941 if(0<uo(i,j))then
942 if(0<va)then !case1
943 ica=1
944 io(1:4)=ijofs(1:4,ica)
945 jo(1:4)=ijofs(5:8,ica)
946 else !case2
947 ica=2
948 io(1:4)=ijofs(1:4,ica)
949 jo(1:4)=ijofs(5:8,ica)
950 end if

```

```

951 else
952   if(0<va)then !case3
953     icode=3
954     io(1:4)=ijofs(1:4,icode)
955     jo(1:4)=ijofs(5:8,icode)
956   else !case4
957     icode=4
958     io(1:4)=ijofs(1:4,icode)
959     jo(1:4)=ijofs(5:8,icode)
960   end if
961 end if
962 !@u (i1,j1),(i2,j2),(i3,j3),(i4,j4) | @v (i1,j1),(i2,j2),(i3,j3),(i4,j4)
963 !va:( 0, 0),( 0, 1),(-1, 1),(-1, 0) | ua:( 0, 0),( 1, 0),( 1,-1),( 0,-1)
964 ! 1:( 0, 0),(-1, 0),( 0, 0),( 0,-1) | 1:( 0, 0),( 0,-1),( 0, 0),(-1, 0)
965 ! 2:( 0, 0),(-1, 0),( 0, 1),( 0, 0) | 2:( 0, 0),( 0,-1),( 1, 0),( 0, 0)
966 ! 3:( 1, 0),( 0, 0),( 0, 0),( 0,-1) | 3:( 0, 1),( 0, 0),( 0, 0),(-1, 0)
967 ! 4:( 1, 0),( 0, 0),( 0, 1),( 0, 0) | 4:( 0, 1),( 0, 0),( 1, 0),( 0, 0)
968 !for u
969 !va=v(i,j)+v(i,j+1)+v(i-1,j+1)+v(i-1,j)
970 !0<u,0<va: u(i,j)*(m(i,j)-m(i-1,j))/dx + va(i,j)*(m(i,j)-m(i,j-1))/dy
971 !0<u,va<0: u(i,j)*(m(i,j)-m(i-1,j))/dx + va(i,j)*(m(i,j+1)-m(i,j))/dy
972 !u<0,0<va: u(i,j)*(m(i+1,j)-m(i,j))/dx + va(i,j)*(m(i,j)-m(i,j-1))/dy
973 !u<0,va<0: u(i,j)*(m(i+1,j)-m(i,j))/dx + va(i,j)*(m(i,j+1)-m(i,j))/dy
974 !for v
975 !ua=u(i,j)+u(i+1,j)+u(i+1,j-1)+u(i,j-1)
976 !0<v,0<ua: v(i,j)*(n(i,j)-n(i,j-1))/dy + ua(i,j)*(n(i,j)-n(i-1,j))/dx
977 !0<v,ua<0: v(i,j)*(n(i,j)-n(i,j-1))/dy + ua(i,j)*(n(i+1,j)-n(i,j))/dx
978 !v<0,0<ua: v(i,j)*(n(i,j+1)-n(i,j))/dy + ua(i,j)*(n(i,j)-n(i-1,j))/dx
979 !v<0,ua<0: v(i,j)*(n(i,j+1)-n(i,j))/dy + ua(i,j)*(n(i+1,j)-n(i,j))/dx
980 x1=uo(i,j)*(uh(i+io(1),j+jo(1))-uh(i+io(2),j+jo(2)))/dx + &
981     va*(uh(i+io(3),j+jo(3))-uh(i+io(4),j+jo(4)))/dx
982 !
983 dhdx=(zo(i,j)+ho(i,j)-zo(i2,j2)-ho(i2,j2))/dx
984 x2=g*h1*dhdx
985 !
986 cost=cos(atan(tan1))
987 tauy=(c1/csta)**f1d5*(sig-rho1)*c1*g*h1*cost*tanp
988 !if(c1<0.05d0)then
989 ! x3=g*rn*rn*uo(i,j)*abs(uo(i,j))/h1**f1d3
990 !else
991 if(c1<0.05d0)then
992   fb=(6.d0+2.5d0*log(h1/dm))**(-2.d0)
993 else
994   fd=kd*sig/rho1*(1.d0-e2)*c1**(f1d3)
995   ff=kf*(1.d0-c1)**f5d3/c1**(f2d3)
996   fb=6.25d0*(fd+ff)*(h1/dm)**(-2.d0)
997   !fb=6.25d0*(fd+ff)*(max(h1,dm)/dm)**(-2.d0)
998   fb2=(6.d0+2.5d0*log(h1/dm))**(-2.d0)
999   if(fb<fb2)fb=fb2
1000 end if
1001 taub=tauy+rho1*fb*(uo(i,j)*uo(i,j)+va*va)
1002 rhom=(sig-rho1)*c1+rho1
1003 !if(uo(i,j)*uo(i,j)+va*va<0.00001d0)then
1004 if(uo(i,j)*uo(i,j)+va*va<g*hlim)then
1005   x3=0.d0

```

```

1006  else
1007      !x3=taub*uo(i,j)/(uo(i,j)*uo(i,j)+va*va)**0.5d0/rhom
1008      x3=taub/(2.d0*rhom*(uo(i,j)*uo(i,j)+va*va)**0.5d0)
1009  end if
1010  !u(i,j)=(uo(i,j)*h1+(-x1-x2-x3)*dt)/hnw
1011  u(i,j)=(uo(i,j)*(h1-x3*dt)-(x1+x2)*dt)/(hnw+x3*dt)
1012  !
1013  if(u(i,j)*uo(i,j)<0.d0)then
1014      u(i,j)=0.d0
1015  end if
1016  htmp(i,j)=hnw
1017  end do
1018  !$omp end parallel do
1019  umax=maxval(abs(u(:,:)))
1020  iloc=maxloc(abs(u(:,:)))
1021  i_umax=iloc(1)
1022  j_umax=iloc(2)
1023  h_umax=htmp(i_umax,j_umax)
1024  !
1025  end subroutine cal_u
1026  !
1027  !
1028  !
1029  subroutine cal_q(n_1d,ij_1d,iend,jend,iofs,jofs,h,u,qx,cc,cf,qccx,qcfx,uh)
1030  use omp_lib
1031  implicit none
1032  integer :: ni,n_1d,ij_1d(n_1d,2),i,iend,j,jend,iofs,jofs
1033  real(8) :: h(iend,jend),u(iend,jend),qx(iend,jend),cc(iend,jend),cf(iend,jend), &
1034           qccx(iend,jend),qcfx(iend,jend),uh(iend,jend)
1035  !
1036  !u:iofs=-1,jofs=0
1037  !v:iofs=0,jofs=-1
1038  !$omp parallel do private(ni,i,j)
1039  do ni=1,n_1d
1040      i=ij_1d(ni,1)
1041      j=ij_1d(ni,2)
1042      uh(i,j)=u(i,j)*(h(i,j)+h(i+iofs,j+jofs))*0.5d0
1043      if(u(i,j)>0.d0)then
1044          qx(i,j)=h(i+iofs,j+jofs)*u(i,j)
1045          qccx(i,j)=qx(i,j)*cc(i+iofs,j+jofs)
1046          qcfx(i,j)=qx(i,j)*cf(i+iofs,j+jofs)*(1.d0-cc(i+iofs,j+jofs))
1047      else
1048          qx(i,j)=h(i,j)*u(i,j)
1049          qccx(i,j)=qx(i,j)*cc(i,j)
1050          qcfx(i,j)=qx(i,j)*cf(i,j)*(1.d0-cc(i,j))
1051      end if
1052  end do
1053  !$omp end parallel do
1054  !
1055  end subroutine cal_q
1056  !
1057  !
1058  !
1059  subroutine cal_qs(n_1d,ij_1d,iend,jend,iofs,jofs,dx,ks,zs,hs,qsx,umax)
1060  use omp_lib

```



```

1061 implicit none
1062 integer :: ni, n_1d, ij_1d(n_1d,2), i, iend, j, jend, iofs, jofs, i1, j1
1063 real(8) :: zs(iend, jend), hs(iend, jend), qsx(iend, jend), &
1064          dx, ks, grad, u, umax, hs1
1065 !
1066 !u:iofs=-1,jofs=-0
1067 !v:iofs= 0,jofs=-1
1068 umax=0.d0
1069 !$omp parallel do private(ni, i, j, i1, j1, grad, hs1, u) reduction(max: umax)
1070 do ni=1, n_1d
1071   i=ij_1d(ni, 1)
1072   j=ij_1d(ni, 2)
1073   i1=i+iofs
1074   j1=j+jofs
1075   grad=(zs(i1, j1)+hs(i1, j1)-(zs(i, j)+hs(i, j)))/dx
1076   if(grad>0.d0)then
1077     hs1=hs(i1, j1)
1078   else
1079     hs1=hs(i, j)
1080   end if
1081   if(hs1<=0.d0)then
1082     qsx(i, j)=0.d0
1083     u=0.d0
1084   else
1085     qsx(i, j)=hs1*grad*ks
1086     u=qsx(i, j)/hs1
1087   end if
1088   if(umax<abs(u))then
1089     umax=abs(u)
1090   end if
1091 end do
1092 !$omp end parallel do
1093 !
1094 end subroutine cal_qs
1095 !
1096 !
1097 !
1098 subroutine cal_hs(n_ij_cv, ij_cv, iend, jend, dt, dx, hs, r, qsx, qsy, csta)
1099 use omp_lib
1100 implicit none
1101 integer :: ni, n_ij_cv, ij_cv(n_ij_cv, 2), i, iend, j, jend
1102 real(8) :: dt, dx, csta, hs(iend, jend), qsx(iend, jend), qsy(iend, jend), r(iend, jend)
1103 !
1104 !$omp parallel do private(ni, i, j)
1105 do ni=1, n_ij_cv
1106   i=ij_cv(ni, 1)
1107   j=ij_cv(ni, 2)
1108   hs(i, j)=hs(i, j) &
1109     - (qsx(i+1, j)-qsx(i, j))/dx*dt/(1.d0-csta) &
1110     - (qsy(i, j+1)-qsy(i, j))/dx*dt/(1.d0-csta) &
1111     + r(i, j)/(1.d0-csta)*dt
1112   if(hs(i, j)<0.d0)then
1113     write(*, '(a, 2i5, E20.10e3)') 'hs<0_□(i, j, h)', i, j, hs(i, j)
1114     hs(i, j)=0.d0
1115   end if

```

```

1116 end do
1117 !$omp end parallel do
1118 !
1119 end subroutine cal_hs
1120 !
1121 !
1122 !
1123 subroutine cal_stability(n_ij_cv, ij_cv, iend, jend, grads, h, hs, lamda, rho, sig, d, tanp, c, g, sf)
1124 use omp_lib
1125 implicit none
1126 real(8), parameter :: PI=acos(-1.0d0), deg2rad=PI/180.0d0, rad2deg=180.0d0/PI
1127 integer :: ni, n_ij_cv, ij_cv(n_ij_cv,2), i, j, iend, jend
1128 real(8) :: lamda, rho, sig, tanp, g, grads(iend, jend), &
1129           h(iend, jend), hs(iend, jend), c(1:iend, 1:jend), sf(iend, 1:jend), pw(iend, jend), &
1130           d(iend, jend), grav, resi, cost, csta, sint, tant
1131 !
1132 csta=1.d0-lamda
1133 !$omp parallel do private(ni, i, j, sint, cost, tant, grav, resi)
1134 do ni=1, n_ij_cv
1135   i=ij_cv(ni,1)
1136   j=ij_cv(ni,2)
1137   sint=sin(atan(grads(i, j)))
1138   cost=cos(atan(grads(i, j)))
1139   tant=grads(i, j)
1140   pw(i, j)=0.d0
1141   !G=rho*g*D*sint*(sig/rho*csta+(1.-hs/D)*pw+hs/D*(1.-csta)+h/D)
1142   !R=rho*g*D*cost*(sig/rho*csta+(1.-hs/D)*pw-hs/D*csta)*tanp+c
1143   if(0.d0<d(i, j))then
1144     grav=rho*g*d(i, j)*sint*(sig/rho*(1.d0-lamda)+(1.d0-hs(i, j)/d(i, j))*pw(i, j)+hs(i, j)/d(i, j)
1145     )*lamda+h(i, j)/d(i, j))
1146     resi=rho*g*d(i, j)*cost*(sig/rho*(1.d0-lamda)+(1.d0-hs(i, j)/d(i, j))*pw(i, j)-hs(i, j)/d(i, j)
1147     )*(1.d0-lamda))*tanp+c(i, j)
1148     sf(i, j)=grav/resi !G/R
1149   else
1150     sf(i, j)=0.d0
1151   end if
1152 !
1153 end do
1154 !$omp end parallel do
1155 !
1156 end subroutine cal_stability
1157 !
1158 subroutine cal_h(n_ij_cv, ij_cv, iend, jend, dt, dx, h, ho, qx, qy, e, csta, cstad)
1159 use omp_lib
1160 implicit none
1161 integer :: ni, n_ij_cv, ij_cv(n_ij_cv,2), i, iend, j, jend
1162 real(8) :: dt, dx, csta, cstad, h(iend, jend), ho(iend, jend), qx(iend, jend), qy(iend, jend), e(iend,
1163 jend)
1164 !$omp parallel do private(ni, i, j)
1165 do ni=1, n_ij_cv
1166   i=ij_cv(ni,1)
1167   j=ij_cv(ni,2)

```

```

1168  if(e(i,j)>=0.d0)then
1169      h(i,j)=ho(i,j)-(qx(i+1,j)-qx(i,j))/dx*dt-(qy(i,j+1)-qy(i,j))/dx*dt+e(i,j)/csta*dt
1170  else
1171      h(i,j)=ho(i,j)-(qx(i+1,j)-qx(i,j))/dx*dt-(qy(i,j+1)-qy(i,j))/dx*dt+e(i,j)/cstad*dt
1172  end if
1173  if(h(i,j)<0.d0)then
1174      write(*,'(a,2i5,4E15.5e3)')'h<0_□(i,j,h,e)',i,j,h(i,j),e(i,j),(qx(i+1,j)-qx(i,j)),(qy(i,j
+1)-qy(i,j))
1175      h(i,j)=0.d0
1176  end if
1177  end do
1178  !$omp end parallel do
1179  !
1180  end subroutine cal_h
1181  !
1182  !
1183  !
1184  subroutine cal_cc(n_ij_cv,ij_cv,iend,jend,dt,dx,hlim,h,ho,cc,cco,qccx,qccy,e,pc)
1185  use omp_lib
1186  implicit none
1187  integer :: ni,n_ij_cv,ij_cv(n_ij_cv,2),i,iend,j,jend
1188  real(8) :: dt,dx,pc,hlim,h(iend,jend),ho(iend,jend),cc(iend,jend),cco(iend,jend), &
1189          qccx(iend,jend),qccy(iend,jend),e(iend,jend)
1190  !
1191  !$omp parallel do private(ni,i,j)
1192  do ni=1,n_ij_cv
1193      i=ij_cv(ni,1)
1194      j=ij_cv(ni,2)
1195      if(h(i,j)<hlim)then
1196          cc(i,j)=0.d0
1197      else
1198          if(e(i,j)>=0.d0)then
1199              cc(i,j)=(cco(i,j)*ho(i,j)-(qccx(i+1,j)-qccx(i,j))/dx*dt-(qccy(i,j+1)-qccy(i,j))/dx*dt+
pc*e(i,j)*dt)/h(i,j)
1200          else
1201              cc(i,j)=(cco(i,j)*ho(i,j)-(qccx(i+1,j)-qccx(i,j))/dx*dt-(qccy(i,j+1)-qccy(i,j))/dx*dt+
e(i,j)*dt)/h(i,j)
1202          end if
1203          if(cc(i,j)<0.d0)then
1204              if(cc(i,j)<-0.001d0)write(*,'(a,2i5,4f15.10)')'cc<0_□(i,j,cco*ho,cc*h,e,dqx+dqy)', &
1205                  i,j,cco(i,j)*ho(i,j),cc(i,j)*h(i,j),e(i,j), &
1206                  (qccx(i+1,j)-qccx(i,j))/dx*dt + (qccy(i,j+1)-qccy(i,j))/dx*dt
1207              !read(*,*)
1208              cc(i,j)=0.d0
1209          end if
1210      end if
1211  end do
1212  !$omp end parallel do
1213  !
1214  end subroutine cal_cc
1215  !
1216  !
1217  !
1218  subroutine cal_cf(n_ij_cv,ij_cv,iend,jend,dt,dx,hlim,h,ho,cf,cfo,cc,cco,qcfx,qcfy,e,pf,cstad
)

```

```

1219 use omp_lib
1220 implicit none
1221 integer :: ni, n_ij_cv, ij_cv(n_ij_cv,2), i, iend, j, jend
1222 real(8) :: dt, dx, pf, cstad, hlim, h(iend, jend), ho(iend, jend), cf(iend, jend), cfo(iend, jend), &
1223         qcfx(iend, jend), qcfy(iend, jend), e(iend, jend), cc(iend, jend), cco(iend, jend)
1224 !
1225 !$omp parallel do private(ni, i, j)
1226 do ni=1, n_ij_cv
1227     i=ij_cv(ni, 1)
1228     j=ij_cv(ni, 2)
1229     if(h(i, j)<hlim)then
1230         cf(i, j)=0.d0
1231     else
1232         if(e(i, j)>=0.d0)then
1233             cf(i, j)=(cfo(i, j)*(1.d0-cco(i, j))*ho(i, j)-(qcfx(i+1, j)-qcfx(i, j))/dx*dt-(qcfy(i, j+1)-
1234                 qcfy(i, j))/dx*dt+pf*e(i, j)*dt) &
1235                 /((1.d0-cc(i, j))*h(i, j))
1236         else
1237             cf(i, j)=(cfo(i, j)*(1.d0-cco(i, j))*ho(i, j)-(qcfx(i+1, j)-qcfx(i, j))/dx*dt &
1238                 -(qcfy(i, j+1)-qcfy(i, j))/dx*dt+((1.d0-cstad)/
1239                 cstad)*cfo(i, j)*e(i, j)*dt) &
1240                 /((1.d0-cc(i, j))*h(i, j))
1241         end if
1242     end if
1243     if(cf(i, j)<0.d0)then
1244         write(*, '(a, 2i5, 4E20.10e3)') 'cc<0 (i, j, cc, cf, h)', i, j, cc(i, j), cf(i, j), e(i, j), h(i, j)
1245         write(*, '(4f15.10)') qcfx(i+1, j), qcfx(i, j), qcfy(i, j+1), qcfy(i, j)
1246         !read(*, *)
1247         cf(i, j)=0.d0
1248     end if
1249 end do
1250 !$omp end parallel do
1251 !
1252 !
1253 !
1254 subroutine cal_e(n_ij_cv, ij_cv, iend, jend, dt, dx, z, zs, sig, rho, csta, tanp, cc, cf, e, h, hlim, ua, va,
1255 &
1256         qx, qy, qccx, qccy, qcfx, qcfy, ifld, tant_e, uv)
1257 use omp_lib
1258 implicit none
1259 real(8), parameter :: PI=acos(-1.d0), deg2rad=PI/180.d0, rad2deg=180.d0/PI
1260 integer :: ni, n_ij_cv, ij_cv(n_ij_cv,2), i, iend, j, jend, ifld(iend, jend)
1261 real(8) :: dt, dx, sig, tanp, tant, tane, csta, emax, dzdx, dzdy, hlim, &
1262         z(iend, jend), zs(iend, jend), cc(iend, jend), cf(iend, jend), e(iend, jend), &
1263         h(iend, jend), rho(iend, jend), wl(iend, jend), &
1264         qx(iend, jend), qy(iend, jend), qccx(iend, jend), qccy(iend, jend), qcfx(iend, jend), qcfy(
1265         iend, jend), &
1266         emin1, emin2, emin3, ua(iend, jend), va(iend, jend), uv(iend, jend), tant_e(iend, jend)
1267 !
1268 !$omp parallel do private(ni, i, j, dzdx, dzdy, tant, tane, &
1269         emax, emin1, emin2, emin3)
1270 do ni=1, n_ij_cv

```

```

1270 i=ij_cv(ni,1)
1271 j=ij_cv(ni,2)
1272 if(h(i,j)<hlim .and. ifld(i,j)==0)then
1273   e(i,j)=0.d0
1274   cycle
1275 end if
1276 if(ua(i,j)>=0.d0)then
1277   dzdx=(wl(i+1,j)-wl(i,j))/dx
1278 else
1279   dzdx=(wl(i,j)-wl(i-1,j))/dx
1280 end if
1281 if(va(i,j)>=0.d0)then
1282   dzdy=(wl(i,j+1)-wl(i,j))/dx
1283 else
1284   dzdy=(wl(i,j)-wl(i,j-1))/dx
1285 end if
1286 if(uv(i,j)==0.d0)then
1287   tant=0.d0
1288 else
1289   !tant=max(0.d0,-(dzdx*ua(i,j)+dzdy*va(i,j))/((ua(i,j)*ua(i,j)+va(i,j)*va(i,j))*0.5d0))
1290   tant=(dzdx*dzdx+dzdy*dzdy)**0.5d0
1291 end if
1292 !tant=(dzdx*dzdx+dzdy*dzdy)**0.5d0
1293 tant_e(i,j)=tant
1294 tane=(sig/rho(i,j)-1.d0)*cc(i,j)*tanp/((sig/rho(i,j)-1.d0)*cc(i,j)+1.d0)
1295 !tan(a-b)=(tan(a)-tan(b))/(1+tan(a)*tan(b))
1296 e(i,j)=csta*(tant-tane)/(1.d0+tant*tane)*uv(i,j)
1297 emax=(z(i,j)-zs(i,j))/dt*csta*cos(atan(tant))
1298 if(e(i,j)>=0.d0)then ! ero
1299   e(i,j)=min(emax,e(i,j))
1300   !if(ifld(i,j)==1)e(i,j)=emax
1301 else ! depo
1302   emin1=min(0.d0,(-h(i,j)/dt*csta+(qx(i+1,j)-qx(i,j))/dx+(qy(i,j+1)-qy(i,j))/dx)*csta) !
1303   cstad -> csta
1304   emin2=min(0.d0,-cc(i,j)*h(i,j)/dt+(qccx(i+1,j)-qccx(i,j))/dx+(qccy(i,j+1)-qccy(i,j))/dx)
1305   if(cf(i,j)>0.d0)then
1306     emin3= &
1307     (-cf(i,j)*(1.d0-cc(i,j))*h(i,j)/dt+(qcfx(i+1,j)-qcfx(i,j))/dx+(qcfy(i,j+1)-qcfy(i,j))/
1308     dx) &
1309     *csta/((1.d0-csta)*cf(i,j))
1310   else
1311     emin3=e(i,j)
1312   end if
1313   emin3=min(0.d0,emin3)
1314   e(i,j)=max(emin1,emin2,emin3,e(i,j))
1315 end if
1316 if(e(i,j)>emax)then
1317   write(*,'(a,5f10.3)')'el<---_max',e(i,j),emax,emin1,emin2,emin3
1318   read(*,*)
1319   e(i,j)=emax
1320 end if
1321 end do
1322 !Somp end parallel do
1323 !
1324 end subroutine cal_e

```

```

1323 !
1324 !
1325 !
1326 subroutine cal_grad(n_ij_cv , ij_cv , iend , jend , dx , z , grad)
1327 use omp_lib
1328 implicit none
1329 real(8) , parameter :: PI=acos(-1.d0) , deg2rad=PI/180.d0 , rad2deg=180.d0/PI
1330 integer :: ni , n_ij_cv , ij_cv(n_ij_cv , 2) , i , iend , j , jend
1331 real(8) :: dx , dx2 , dzdx , dzdy , z(iend , jend) , grad(iend , jend)
1332 !
1333 dx2=dx*2.d0
1334 !do j=2,jend-1
1335 !do i=2,iend-1
1336 !$omp parallel do private(ni , i , j , dzdx , dzdy)
1337 do ni=1 , n_ij_cv
1338   i=ij_cv(ni , 1)
1339   j=ij_cv(ni , 2)
1340   dzdx=(z(i+1 , j)-z(i-1 , j))/dx2
1341   dzdy=(z(i , j+1)-z(i , j-1))/dx2
1342   grad(i , j)=(dzdx*dzdx+dzdy*dzdy)**0.5d0
1343 end do
1344 !$omp end parallel do
1345 !
1346 end subroutine cal_grad
1347 !
1348 !
1349 !
1350 subroutine cal_z(n_ij_cv , ij_cv , iend , jend , dt , csta , z , zs , e , tant_e)
1351 use omp_lib
1352 implicit none
1353 real(8) , parameter :: PI=acos(-1.d0) , deg2rad=PI/180.d0 , rad2deg=180.d0/PI
1354 integer :: ni , n_ij_cv , ij_cv(n_ij_cv , 2) , i , iend , j , jend
1355 real(8) :: dt , csta , tant , cost , z(iend , jend) , zs(iend , jend) , e(iend , jend) , tant_e(iend , jend)
1356 !
1357 !$omp parallel do private(ni , i , j , tant , cost)
1358 do ni=1 , n_ij_cv
1359   i=ij_cv(ni , 1)
1360   j=ij_cv(ni , 2)
1361   tant=tant_e(i , j)
1362   cost=cot(atan(tant))
1363   z(i , j)=z(i , j)-e(i , j)/csta/cost*dt
1364   if(z(i , j)<zs(i , j))then
1365     if(z(i , j)-zs(i , j)<-0.0001d0)write(* , '(a,2i5,5f15.5)')'z<_zs@(i , j)=' , i , j , e(i , j) , z(i , j)
1366     z(i , j)=zs(i , j)
1367   end if
1368 end do
1369 !$omp end parallel do
1370 !
1371 end subroutine cal_z
1372 !
1373 !
1374 !
1375 subroutine cal_rho(n_ij_cv , ij_cv , iend , jend , sig , rhow , cc , cf , rho , rhom)
1376 use omp_lib

```

```

1377 implicit none
1378 integer :: ni, n_ij_cv, ij_cv(n_ij_cv,2), i, iend, j, jend
1379 real(8) :: sig, rhow, cc(iend, jend), cf(iend, jend), rho(iend, jend), rhom(iend, jend)
1380 !
1381 !$omp parallel do private(ni, i, j)
1382 do ni=1, n_ij_cv
1383   i=ij_cv(ni, 1)
1384   j=ij_cv(ni, 2)
1385   rho(i, j)=rhow+cf(i, j)*(sig-rhow)
1386   rhom(i, j)=rho(i, j)+cc(i, j)*(sig-rho(i, j))
1387 end do
1388 !$omp end parallel do
1389 !
1390 end subroutine cal_rho
1391 !
1392 !
1393 !
1394 subroutine r_iasc(fon, fname, ia, iend, jend, xllcorner, yllcorner, cellsize)
1395 implicit none
1396 integer :: j, iend, jend, fon, ncols, nrows, ia(1:iend, 1:jend)
1397 real(8) :: xllcorner, yllcorner, cellsize
1398 character(len=100) :: ctmp, fname
1399 !
1400 open(fon, file=trim(adjustl(fname)))
1401 read(fon, *) ctmp, ncols
1402 read(fon, *) ctmp, nrows
1403 if (ncols /= iend .or. nrows /= jend) then
1404   write(*, *) 'Check_ncols_/=_iend_or_nrows_/=_jend'
1405   stop
1406 end if
1407 read(fon, *) ctmp, xllcorner
1408 read(fon, *) ctmp, yllcorner
1409 read(fon, *) ctmp, cellsize
1410 read(fon, *)
1411 do j=jend, 1, -1
1412   read(fon, *) ia(1:iend, j)
1413 end do
1414 close(fon)
1415 !
1416 end subroutine r_iasc
1417 !
1418 !
1419 !
1420 subroutine r_fasc(fon, fname, a, iend, jend, xllcorner, yllcorner, cellsize)
1421 implicit none
1422 integer :: j, iend, jend, fon, ncols, nrows
1423 real(8) :: xllcorner, yllcorner, cellsize, a(1:iend, 1:jend)
1424 character(len=100) :: ctmp, fname
1425 !
1426 open(fon, file=trim(adjustl(fname)))
1427 read(fon, *) ctmp, ncols
1428 read(fon, *) ctmp, nrows
1429 if (ncols /= iend .or. nrows /= jend) then
1430   write(*, *) 'Check_ncols_/=_iend_or_nrows_/=_jend'
1431   stop

```

```

1432 end if
1433 read(fon,*)ctmp,xllcorner
1434 read(fon,*)ctmp,yllcorner
1435 read(fon,*)ctmp,cellsize
1436 read(fon,*)
1437 do j=jend,1,-1
1438   read(fon,*)a(1:iend,j)
1439 end do
1440 close(fon)
1441 !
1442 end subroutine r_fasc
1443 !
1444 !
1445 !
1446 subroutine w_fasc_fmt(fon,fname,a,str,iend,jend,xllcorner,yllcorner,cellsize,nv)
1447 implicit none
1448 integer :: j,iend,jend,fon
1449 real(8) :: a(1:iend,1:jend),xllcorner,yllcorner,cellsize,nv
1450 character(len=100) :: fname,ciend,str
1451 !
1452 write(ciend,*)iend
1453 open(fon,file=trim(adjustl(fname)))
1454 write(fon,'(a,i5)')'ncols',iend
1455 write(fon,'(a,i5)')'nrows',jend
1456 write(fon,'(a,f15.3)')'xllcorner',xllcorner
1457 write(fon,'(a,f15.3)')'yllcorner',yllcorner
1458 write(fon,'(a,f10.3)')'cellsize_',cellsize
1459 write(fon,'(a,f10.3)')'NODATA_value',nv
1460 do j=jend,1,-1
1461   write(fon,trim(adjustl(str)))a(1:iend,j)
1462 end do
1463 close(fon)
1464 !
1465 end subroutine w_fasc_fmt
1466 !
1467 !
1468 !
1469 subroutine w_iasc_fmt(fon,fname,ia,str,iend,jend,xllcorner,yllcorner,cellsize,nv)
1470 implicit none
1471 integer :: j,iend,jend,fon,nv,ia(1:iend,1:jend)
1472 real(8) :: xllcorner,yllcorner,cellsize
1473 character(len=100) :: fname,ciend,str
1474 !
1475 write(ciend,*)iend
1476 open(fon,file=trim(adjustl(fname)))
1477 write(fon,'(a,i5)')'ncols',iend
1478 write(fon,'(a,i5)')'nrows',jend
1479 write(fon,'(a,f15.3)')'xllcorner',xllcorner
1480 write(fon,'(a,f15.3)')'yllcorner',yllcorner
1481 write(fon,'(a,f10.3)')'cellsize_',cellsize
1482 write(fon,'(a,i5)')'NODATA_value',nv
1483 do j=jend,1,-1
1484   write(fon,trim(adjustl(str)))ia(1:iend,j)
1485 end do
1486 close(fon)

```



1487 !

1488 **end subroutine** w\_iasc\_fmt

---

## 5 05\_MK\_FIG

RR\_out\_hyeto.py

```

1 import os
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import datetime
5 from matplotlib.dates import DateFormatter
6 #####
7 r_gold=(1.+5.**0.5)*0.5
8 r_silver=(1.+2.**0.5)
9 r_yamato=2.**0.5
10 w_inchi=4
11 h_inchi=w_inchi/r_gold*0.8
12 r_top=0
13 r_bottom=0
14 r_left=0
15 r_right=0
16 fsize=7.5
17 #####
18 #plt.figure(figsize=(w_inchi,h_inchi))
19 plt.rcParams["font.size"]=fsize
20 from matplotlib.font_manager import FontProperties
21 fp=FontProperties(fname=r'C:\WINDOWS\Fonts\msgothic.ttc')
22 #####
23 fig,ax=plt.subplots(1,1,figsize=(w_inchi,h_inchi))
24 ax.xaxis.set_major_formatter(DateFormatter('%b%d\n%H:%M\n'))
25 #ax.xaxis.set_major_formatter(DateFormatter('%H'))
26 ax2=ax.twinx()
27 ax2.xaxis.set_major_formatter(DateFormatter('%b.%d\n%H:%M'))
28 ls='\n'
29 fs=','
30 tdir=os.path.join('.', '04_EXE_SIMU', '01_RR_DR')
31 fn=os.path.join(tdir, 'output_misc', 'basin_rain.txt')
32 txt=np.loadtxt(fn, dtype=str, delimiter=',', skiprows=1)
33 date=txt.T[0]
34 hyet=txt.T[1].astype(float)
35 time1=[]
36 for line in txt.T[0]:
37     date=line.split()[0].split('/')
38     year,month,day=int(date[0]),int(date[1]),int(date[2])
39     time=line.split()[1].split(':')
40     hour,minute=int(time[0]),int(time[1])
41     time0=datetime.datetime(year,month,day,hour,minute)
42     time1.append(time0)
43 dt=time1[1]-time1[0]
44 ax.fill_between(time1,hyet,linewidth=0.,facecolor='gray',alpha=1.,step='pre')
45 ax.plot(datetime.datetime(1990,7,2,9,20),80,marker='^',color='k')
46 arain=[sum(hyet[0:i]) for i in range(1,len(hyet)+1)]
47 ax2.plot(time1,arain,lw=0.8,ls='-',color='k')
48 ax2.set_ylim(0,500)
49 #plt.xlim(datetime.datetime(2017,7,5,0),datetime.datetime(2017,7,6,12))
50 ax.set_xlabel(u'Year_(1990)',fontsize=fsize)

```

```
51 plt.xticks(fontsize=fontsize*0.8)
52 ax.set_ylim(100,0)
53 #ax.set_ylabel(u'%d分雨量 (mm)'%int(dt.seconds/60), fontproperties=fp, fontsize
    =fontsize)
54 #ax2.set_ylabel(u'累積雨量 (mm)', fontproperties=fp, fontsize=fontsize)
55 ax.set_ylabel(u'%d_mih. rainfall (mm) '%int(dt.seconds/60), fontsize=fontsize)
56 ax2.set_ylabel(u'Accumulated rainfall (mm)', fontsize=fontsize)
57 #plt.legend(fontsize=fontsize)
58 plt.tight_layout()
59 #plt.grid()
60 #plt.yscale("log")
61 plt.savefig('RR_hyeto.png', dpi=300)
62 #plt.show()
```

---

```

1 import os
2 import glob
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import matplotlib.cm as cm
6 import datetime
7 from matplotlib.dates import DateFormatter
8 import sys
9 #####
10 r_gold=(1.+5.**0.5)*0.5
11 r_silver=(1.+2.**0.5)
12 r_yamato=2.**0.5
13 w_inchi=4
14 h_inchi=w_inchi/r_gold*0.8
15 r_top=0
16 r_bottom=0
17 r_left=0
18 r_right=0
19 fsize=7.5
20 #####
21 #plt.figure(figsize=(w_inchi, h_inchi))
22 plt.rcParams["font.size"]=fsize
23 from matplotlib.font_manager import FontProperties
24 fp = FontProperties(fname=r'C:\WINDOWS\Fonts\msgothic.ttc')
25 #####
26 fig, ax = plt.subplots(1, 1, figsize=(w_inchi, h_inchi))
27 ax.xaxis.set_major_formatter(DateFormatter('%b%d\n%H:%M\n'))
28 #ax.xaxis.set_major_formatter(DateFormatter('%H'))
29 # fname='Terauchi_dam.csv'
30 # dt=3600
31 # data=np.loadtxt(fname, dtype=str, delimiter=',', skiprows=0)
32 # iskp=0
33 # q_obs=np.transpose(data)[1].astype('float')
34 # cont=np.transpose(data)[0][0]
35 # date, time=cont.split()[0].split('/'), cont.split()[1].split(':')
36 # year, month, day=int(date[0]), int(date[1]), int(date[2])
37 # hour, minute=int(time[0]), int(time[1])
38 # time0=datetime.datetime(year, month, day, hour, minute)
39 # time0=time0-datetime.timedelta(seconds=0)
40 # time=[time0+datetime.timedelta(seconds=dt*i) for i in range(len(q_obs))]
41 # plt.plot(time, q_obs, lw=0.6, ls='--', color='k', label=r'obs.')
42 #
43 # #####
44 ls='\n'
45 dirs=glob.glob(os.path.join('.', '04_EXE_SIMU', '01_RR_DR'))
46 #loc=[4575,4741] # 343:St.1, 1574 1562:st.7
47 loc=sys.argv[1:]
48 loc=[int(s) for s in loc]
49 print('out_@', loc)
50 model='RR'
51 lgs=[[1] for i in range(len(dirs))]
52 for lg, tdir in zip(lgs, dirs):

```

```

52     if model=='RR': files=glob.glob(os.path.join(tdir, 'output_RR', 'res_1d_*.
txt'))
53     files.sort()
54     iend=len(files)
55     label=tdir.split('_')[-1]
56     fn=os.path.join(tdir, 'output_misc', 'basin_rain.txt')
57     cont=open(fn).read().split('\n')[1].split(',')[0].strip(' ')
58     date, time=cont.split()[0].split('/'), cont.split()[1].split(':')
59     year, month, day=int(date[0]), int(date[1]), int(date[2])
60     hour, minute=int(time[0]), int(time[1])
61     time0=datetime.datetime(year, month, day, hour, minute)
62     time0=time0-datetime.timedelta(seconds=0)
63     hydro, hhh, uuu = [], [], []
64     qdf, pcc, pcf = [], [], []
65     qa, qca, qfa = [], [], []
66     time=[]
67     for i0, afile in enumerate(files[:]):
68         res=np.loadtxt(afile, usecols=(3,4,5,6))
69         sec=float(os.path.basename(afile).split('_')[2][: -4])
70         #print(afile)
71         z=np.transpose(res)[0]
72         h=np.transpose(res)[1]
73         u=np.transpose(res)[2]
74         q=np.transpose(res)[3]
75         hydro.append([q[l-1] for l in loc])
76         hhh.append([h[l-1] for l in loc])
77         uuu.append([u[l-1] for l in loc])
78         time.append(time0+datetime.timedelta(seconds=sec))
79         #print(sec, time[-1])
80     for i in range(len(loc)):
81         plt.plot(time, np.transpose(hydro)[i], lw=lg[0], marker='', label='No.%d'
%loc[i])
82 plt.xlim(datetime.datetime(1990,7,2,8), datetime.datetime(1990,7,2,14))
83 plt.xlabel(u'Time/Date_(1999)', fontsize=fsize)
84 plt.ylabel(u'Discharge_(m3/s)', fontsize=fsize)
85 plt.ylim(0,)
86 plt.legend(fontsize=fsize)
87 #plt.grid()
88 plt.tight_layout()
89 plt.savefig('%s_hydro.png'%model, dpi=300)
90 #plt.show()

```

---

```

1  # -*- coding: utf-8 -*-
2  import os
3  import glob
4  import numpy as np
5  import matplotlib.pyplot as plt
6  import matplotlib.cm as cm
7  import datetime
8  from matplotlib.dates import DateFormatter
9  from matplotlib.ticker import ScalarFormatter
10 import sys
11 #####
12 r_gold=(1.+5.**0.5)*0.5
13 r_silver=(1.+2.**0.5)
14 r_yamato=2.**0.5
15 w_inchi=3.1496
16 h_inchi=w_inchi
17 r_top=0
18 r_bottom=0
19 r_left=0
20 r_right=0
21 fsize=7
22 #####
23 #plt.figure(figsize=(w_inchi, h_inchi))
24 plt.rcParams["font.size"] = fsize
25 from matplotlib.font_manager import FontProperties
26 fp = FontProperties(fname=r'C:\WINDOWS\Fonts\msgothic.ttc')
27 #####
28 fig, ax = plt.subplots(3,1, figsize=(w_inchi, h_inchi))
29 for i in range(3):
30     ax[i].xaxis.set_major_formatter(DateFormatter('%b.%d\n%H:%M'))
31 ax2=[ax[i].twinx() for i in range(3)]
32 [ax2ax.xaxis.set_major_formatter(DateFormatter('%b.%d\n%H:%M')) for ax2ax in
    ax2]
33 ls='\n'
34 dirs=glob.glob(os.path.join('.', '04_EXE_SIMU', '01_RR_DR'))
35 loc=sys.argv[1:]
36 loc=[int(s) for s in loc]
37 print('out_@', loc)
38 lgs=[[1] for i in range(len(dirs))]
39 for lg, tdir in zip(lgs, dirs):
40     files=glob.glob(os.path.join(tdir, 'output_DR', 'res_*.txt'))
41     files.sort()
42     iend=len(files)
43     label=tdir.split('_')[-1]
44     #fn=os.path.join(tdir, 'RR_input.txt')
45     #cont=open(fn).read().split(ls)[8].split()[0]
46     #rdt=float(cont.replace('d0', '0'))
47     fn=os.path.join(tdir, 'output_misc', 'basin_rain.txt')
48     cont=open(fn).read().split(ls)[1].split(',')[0].strip(" ")
49     date, time=cont.split()[0].split('/'), cont.split()[1].split(':')
50     year, month, day=int(date[0]), int(date[1]), int(date[2])
51     hour, minute=int(time[0]), int(time[1])
52     time0=datetime.datetime(year, month, day, hour, minute)

```

```

53     time0=time0-datetime.timedelta(seconds=0)
54     hydro , hhh , uu = [] , [] , []
55     qdf , pcc , pcf = [] , [] , []
56     qa , qca , qfa = [] , [] , []
57     time = []
58     for i0 , afile in enumerate( files [ : ] ):
59         #print ( afile )
60         res=np.loadtxt( afile , usecols=(4,5,6,16,17,18) , skiprows=1)
61         sec=float( os.path.basename( afile ).split( '-' )[-1][ : -4 ])
62         #
63         q=np.transpose( res )[0]
64         cc=np.transpose( res )[1]
65         cf=np.transpose( res )[2]
66         qdf.append( [q[l-1] for l in loc] )
67         pcc.append( [cc[l-1] for l in loc] )
68         pcf.append( [cf[l-1] for l in loc] )
69         #
70         qaa=np.transpose( res )[3]
71         qcaa=np.transpose( res )[4]
72         qfaa=np.transpose( res )[5]
73         qa.append( [qaa[l-1] for l in loc] )
74         qca.append( [qcaa[l-1] for l in loc] )
75         qfa.append( [qfaa[l-1] for l in loc] )
76         #
77         time.append( time0+datetime.timedelta(seconds=sec) )
78     for i in range( len( loc ) ):
79         dt=(time[1]-time[0]).seconds
80         qa , qca , qfa=np.array( qa ) , np.array( qca ) , np.array( qfa )
81         q=np.append( [0] , ( qa.T[i][1:] - qa.T[i][: -1] ) / dt )
82         qc=np.append( [0] , ( qca.T[i][1:] - qca.T[i][: -1] ) / dt )
83         qf=np.append( [0] , ( qfa.T[i][1:] - qfa.T[i][: -1] ) / dt )
84         #
85         ax[0].plot( time , q , lw=0.5 , color='k' , ls='-' , marker='' , label='water_&_
sed.' )
86         ax2[0].plot( time , qa.T[i] , lw=0.5 , color='k' , ls='—' , marker='' , label='No
.%d'%loc[i] )
87         ax[1].plot( time , qc , lw=0.5 , color='k' , ls='-' , marker='' , label='coarse_
sed.' )
88         ax2[1].plot( time , qca.T[i] , lw=0.5 , color='k' , ls='—' , marker='' , label='
No.%d'%loc[i] )
89         ax[2].plot( time , qf , lw=0.5 , color='k' , ls='-' , marker='' , label='fine_sed.
' )
90         ax2[2].plot( time , qfa.T[i] , lw=0.5 , color='k' , ls='—' , marker='' , label='
No.%d'%loc[i] )
91     ti=['(a)' , '(b)' , '(c)']
92     for i in range( 3 ):
93         ax[i].set_xlim( datetime.datetime(1990,7,2,8) , datetime.datetime
(1990,7,2,14) )
94         if i==1: ax[i].set_ylabel( u'Discharge_(m$^3$/s)' , fontsize=fsize )
95         if i<2: ax[i].axes.xaxis.set_ticklabels( [] )
96         ax[i].set_ylim( 0 , )
97         ax[i].legend( loc='center_right' , fontsize=fsize )
98         #ax2=ax[i].twinx()
99         #ax2[i].yaxis.set_major_formatter(ScalarFormatter(useMathText=True))
100        ax2[i].ticklabel_format( style='sci' , axis='y' , scilimits=(0,0) )

```

```
101     ax2[i].set_ylim(0,)
102     if i==1: ax2[i].set_ylabel(u'Volume_(m$^3$)', fontsize=fsz)
103     ax2[i].annotate('%s'%ti[i], xy=[0., 0.], xytext=[0.02,0.79], xycoords='
axes__fraction')
104 #ax[0].axes.xaxis.set_ticklabels([])
105 #ax[1].axes.xaxis.set_ticklabels([])
106 #ax3=ax[2].twinx()
107 #ax3.xaxis.set_major_formatter(DateFormatter('%b.%d\n%H:%M'))
108 ax[2].set_xlabel(u'Date/Time_(1999)', fontsize=fsz)
109 #plt.grid()
110 plt.tight_layout()
111 plt.savefig('DR_hydro.png', dpi=300)
112 #plt.show()
```

---



## 5.1 mk\_map

map.py

```
1 import os
2 import numpy as np
3 import glob
4 import matplotlib.pyplot as plt
5 from matplotlib.colors import Normalize
6 ls='\n'
7 tdir=os.path.join('..','..','04_EXE_SIMU','01_RR_DR')
8 files=glob.glob(os.path.join(tdir,'output_DR','res_?????????.txt'))
9 files.sort()
10 fn=os.path.join(tdir,'input','targetArea_inRR.asc')
11 head=open(fn).read().split(ls)[:6]
12 xllcorner=float(head[2].split())[1]
13 yllcorner=float(head[3].split())[1]
14 dx=float(head[4].split())[1]
15 dy=dx
16 head=ls.join(head)
17 #
18 ij2d=np.loadtxt(fn,skiprows=6)
19 ij2d=np.full(np.shape(ij2d),0,dtype=int)
20 jend,iend=np.shape(ij2d)
21 x=np.linspace(xllcorner,xllcorner+iend*dx,iend)
22 y=np.linspace(yllcorner,yllcorner+jend*dy,jend)
23 Xg,Yg=np.meshgrid(x,y)
24 fn=os.path.join(tdir,'input','streamConfiguration_inRR.txt')
25 idxs=np.loadtxt(fn,skiprows=2,delimiter=',',dtype=int,usecols=(14,15))
26 i_id=np.transpose(idxs)[0]
27 j_id=np.transpose(idxs)[1]
28 #
29 a=np.full(np.shape(ij2d),0,dtype=float)
30 qmax=np.full(np.shape(ij2d),0,dtype=float)
31 umax=np.full(np.shape(ij2d),0,dtype=float)
32 hmax=np.full(np.shape(ij2d),0,dtype=float)
33 dze=np.full(np.shape(ij2d),0,dtype=float)
34 ifld=np.full(np.shape(ij2d),0,dtype=int)
35 hsc=np.full(np.shape(ij2d),0,dtype=float)
36 for fn in files[0:]:
37     print(fn)
38     ucols=tuple(range(17))
39     data=np.loadtxt(fn,skiprows=1,usecols=ucols)
40     za=np.transpose(data)[1]
41     ha=np.transpose(data)[2]
42     ua=np.transpose(data)[3]
43     zinia=np.transpose(data)[9]
44     dza=za-zinia
45     iflda=np.transpose(data)[15]
46     hsca=np.transpose(data)[14]
47     qa=np.transpose(data)[4]
48     dero=np.sum(np.where(dza<0,dza,0))
49     ddep=np.sum(np.where(dza>0,dza,0))
50     #print('%10.1f%10.1f%10.1f'%(dero,ddep,ddep-dero))
```

```

51  #for i,j,h,u,dz,hs,qs,ifld0,hsc0,q in zip(i_id,j_id,ha,ua,dza,hsa,gsa,
    iflda,hsca,qa):
52  for i,j,h,u,dz,ifld0,hsc0,q in zip(i_id,j_id,ha,ua,dza,iflda,hsca,qa):
53      if i==0:
54          print(i,j)
55          continue
56          umax[jend-j,i-1]=max(u,umax[jend-j,i-1])
57          hmax[jend-j,i-1]=max(h,hmax[jend-j,i-1])
58          dze[jend-j,i-1]=dz
59          ifld[jend-j,i-1]=max(ifld0,ifld[jend-j,i-1])
60          hsc[jend-j,i-1]=max(hsc0,hsc[jend-j,i-1])
61          qmax[jend-j,i-1]=max(q,qmax[jend-j,i-1])
62  isec=int(os.path.basename(fn)[4:-4])
63  if isec%36000==0:
64      levels = [-1,0,1]
65      fign=os.path.join('pic','dz_%s.png'%(os.path.basename(fn)[4:-4]))
66      plt.pcolor(Xg,Yg[:,:-1],dze,cmap='coolwarm',norm=Normalize(vmin=-1,
vmax=1))
67      plt.xlabel('East-West_(m)')
68      plt.ylabel('South-North_(m)')
69      plt.colorbar(label='dz_(m)')
70      plt.axes().set_aspect(dx/dy)
71      plt.savefig(fign)
72      plt.close()
73      fign=os.path.join('asc','dz_%s.asc'%(os.path.basename(fn)[4:-4]))
74      np.savetxt(fign,dze,fmt='%7.3f',header=header,comments='')
75  np.savetxt('umax.asc',umax,fmt='%f',header=header,comments='')
76  np.savetxt('hmax.asc',hmax,fmt='%7.2f',header=header,comments='')
77  np.savetxt('dz.asc',dze,fmt='%6.2f',header=header,comments='')
78  np.savetxt('qmax.asc',qmax,fmt='%f',header=header,comments='')
79  #raw_input()

```

---

## 5.2 mk\_vtk

vtk\_line\_RR.py

```

1 import os
2 import numpy as np
3 import io
4 import glob
5 import sys
6 #
7 dt_out=sys.argv[1]
8 dt_out=int(sys.argv[1])
9 print('dt_out=%s'%dt_out)
10 #
11 ls='\n'
12 tpath=os.path.join('..','..','04_EXE_SIMU','01_RR_DR')
13 fo_h='RR_{0:010d}.vtk'
14 files=glob.glob(os.path.join(tpath,'output_RR','res_1d_?????????.txt'))
15 #
16 fvtk=open('stream_line.vtk','w')
17 fn=os.path.join(tpath,'input','streamConfiguration_inRR.txt')
18 flgs=np.loadtxt(fn,skiprows=2,delimiter=',',dtype=str)
19 xa=flgs.T[12].astype(float)
20 ya=flgs.T[13].astype(float)
21 za=flgs.T[ 5].astype(float)
22 ba=flgs.T[ 9].astype(float)
23 acc=flgs.T[11].astype(int)
24 acc=np.log10(acc)
25 na=flgs.T[ 0].astype(int)
26 npo=len(na)
27 head=['#_vtk_DataFile_Version_2.0',
28        'Line_example','ASCII','DATASET_POLYDATA','POINTS_%d_float'%npo]
29 io_head = io.StringIO()
30 io_head.write(ls.join(head)+ls)
31 for x,y,z in zip(xa,ya,za):
32     io_head.write('%f%f%f%s'%(x,y,z,ls))
33 na2=flgs.T[1].astype(int)
34 nli=np.size(np.where(na2!=0))
35 io_head.write('LINES_%d_%d%s'%(nli,3*nli,ls))
36 for i,i2 in zip(na,na2):
37     if 0<i2:
38         io_head.write('%d_%d_%d%s'%(2,i-1,i2-1,ls))
39 head=['CELL_DATA_%d'%nli,'SCALARS_%s_%s_1'%(z,'float'),
40        'LOOKUP_TABLE_default']
41 io_head.write(ls.join(head)+ls)
42 itr=1
43 for i,i2 in zip(na,na2):
44     if 0<i2:
45         z=za[i-1]
46         if itr%10==0:
47             io_head.write('%f%s'%(z,ls))
48         else:
49             io_head.write('%f '%(z))
50     itr=itr+1

```

```

51 for fn in files:
52     sec=int(os.path.basename(fn).split('-')[-1][4:-4])
53     if sec%dt_out==0:
54         data=np.loadtxt(fn, skiprows=0, dtype=str)
55         za=data.T[3].astype(float)
56         ha=data.T[4].astype(float)
57         ua=data.T[5].astype(float)
58         qa=data.T[6].astype(float)
59         fvtk=fo_h.format(sec)
60         fvtk=os.path.join('out', fvtk)
61         print('output_————>', fvtk)
62         vtk=open(fvtk, 'w')
63         vtk.write(io_head.getvalue())
64         for cvar, var in [['h', ha], ['u', ua], ['q', qa], ['z', za]]:
65             vtk.write(ls)
66             head=['SCALARS_%s_%s_1'%(cvar, 'float'), 'LOOKUP_TABLE_default']
67             vtk.write(ls.join(head)+ls)
68             itr=1
69             for i, i2 in zip(na, na2):
70                 if 0<i2:
71                     v=var[i-1]
72                     if itr%10==0:
73                         vtk.write('%f%s'%(v, ls))
74                     else:
75                         vtk.write('%f_'%(v))
76                 itr=itr+1
77         vtk.close()
78 io_head.close()

```

---

```

1 import os
2 import numpy as np
3 import io
4 import glob
5 import sys
6 #
7 dt_out=sys.argv[1]
8 dt_out=int(sys.argv[1])
9 print('dt_out==',dt_out)
10 #
11 ls='\n'
12 tpath=os.path.join('..','..','04_EXE_SIMU','01_RR_DR')
13 fo_h='DR_{0:010d}.vtk'
14 files=glob.glob(os.path.join(tpath,'output_DR','res_?????????.txt'))
15 #
16 fvtk=open('stream_line.vtk','w')
17 fn=os.path.join(tpath,'input','streamConfiguration_inRR.txt')
18 flgs=np.loadtxt(fn,skiprows=2,delimiter=',',dtype=str)
19 xa=flgs.T[12].astype(float)
20 ya=flgs.T[13].astype(float)
21 za=flgs.T[ 5].astype(float)
22 ba=flgs.T[ 9].astype(float)
23 acc=flgs.T[ 11].astype(int)
24 acc=np.log10(acc)
25 na=flgs.T[ 0].astype(int)
26 npo=len(na)
27 head=['#_vtk_DataFile_Version_2.0',
28       'Line_example','ASCII','DATASET_POLYDATA','POINTS_%d_float'%npo]
29 io_head = io.StringIO()
30 io_head.write(ls.join(head)+ls)
31 for x,y,z in zip(xa,ya,za):
32     io_head.write('%f_%f_%f%s'%(x,y,z,ls))
33 na2=flgs.T[1].astype(int)
34 nli=np.size(np.where(na2!=0))
35 io_head.write('LINES_%d_%d%s'%(nli,3*nli,ls))
36 for i,i2 in zip(na,na2):
37     if 0<i2:
38         io_head.write('%d_%d_%d%s'%(2,i-1,i2-1,ls))
39 head=['CELL_DATA_%d'%nli,'SCALARS_%s_%s_1'%(z,'float'),
40       'LOOKUP_TABLE_default']
41 io_head.write(ls.join(head)+ls)
42 itr=1
43 for i,i2 in zip(na,na2):
44     if 0<i2:
45         z=za[i-1]
46         if itr%10==0:
47             io_head.write('%f%s'%(z,ls))
48         else:
49             io_head.write('%f_'%(z))
50         itr=itr+1
51 for fn in files:
52     sec=int(os.path.basename(fn).split('_')[-1][4:-4])
53     if sec%dt_out==0:

```

```

54     data=np.loadtxt(fn,skiprows=1,dtype=str)
55     za=data.T[1].astype(float)
56     ha=data.T[2].astype(float)
57     ua=data.T[3].astype(float)
58     qa=data.T[4].astype(float)
59     cca=data.T[5].astype(float)
60     cfa=data.T[6].astype(float)
61     z0a=data.T[9].astype(float)
62     dza=za-z0a
63     #fvtk=os.path.basename(fn)[-4]+' vtk '
64     fvtk=fo.h.format(sec)
65     fvtk=os.path.join('out',fvtk)
66     print('output————>',fvtk)
67     vtk=open(fvtk,'w')
68     vtk.write(io_head.getvalue())
69     for cvar,var in [['h',ha],['u',ua],['q',qa],['cc',cca],['cf',cfa],['
dz',dza]]:
70         vtk.write(1s)
71         head=['SCALARS_%s_%s_1'%(cvar,'float'),'LOOKUP_TABLE_default']
72         vtk.write(1s.join(head)+1s)
73         itr=1
74         for i,i2 in zip(na,na2):
75             if 0<i2:
76                 v=var[i-1]
77                 if itr%10==0:
78                     vtk.write('%f%s'%(v,1s))
79                 else:
80                     vtk.write('%f_'%(v))
81             itr=itr+1
82     vtk.close()
83 io_head.close()

```

---

```

1 import os
2 import numpy as np
3 import io
4 import glob
5 import sys
6 #
7 dt_out=sys.argv[1]
8 dt_out=int(sys.argv[1])
9 print('dt_out==',dt_out)
10 #
11 ls='\n'
12 tpath=os.path.join('..','..','04_EXE_SIMU','02_DF_small_shift')
13 varis=('h','cc','dz')
14 fo_h='DF_{:010d}.vtk'
15 fz=os.path.join(tpath,'input','elevation_inDF.asc')
16 fb=os.path.join(tpath,'input','targetArea_inDF.asc')
17 files0=glob.glob(os.path.join(tpath,'output_DF','res_*_?????????.asc'))
18 #
19 z=np.loadtxt(fz,skiprows=6,dtype=float)
20 basin=np.loadtxt(fb,skiprows=6,dtype=int)
21 for var in varis:
22     files=[fn for fn in files0 if '_%s_'%var in os.path.basename(fn)]
23     for fn in files:
24         sec=int(os.path.basename(fn).split('_')[-1][4:-4])
25         if sec%dt_out==0:
26             #fvtk=os.path.basename(fn)[-4]+' .vtk'
27             fvtk=fo_h.format(var,sec)
28             fvtk=os.path.join('out',fvtk)
29             print('output ->',fvtk)
30             vtk=open(fvtk,'w')
31             head=open(fn).read().split(ls)[:6]
32             ncols,nrows=int(head[0].split()[1]),int(head[1].split()[1])
33             xco,yco=float(head[2].split()[1]),float(head[3].split()[1])
34             cs=float(head[4].split()[1])
35             #
36             h=np.loadtxt(fn,skiprows=6,dtype=float)
37             head=['#_vtk_DataFile_Version_3.0','cell',
38                 'ASCII','DATASET_STRUCTURED_POINTS',
39                 'DIMENSIONS_%d_%d_%d'%(ncols,nrows,1),
40                 'SPACING_%f_%f_%f'%(cs,cs,0),
41                 #'ORIGIN %f %f %f'%(xco+0.5*cs,yco+0.5*cs,0),
42                 'ORIGIN_%f_%f_%f'%(xco,yco,0),
43                 'POINT_DATA_%d'%(ncols*nrows),
44                 'SCALARS_%s_%s_1'%(z,'float'),
45                 'LOOKUP_TABLE_default']
46             vtk.write(ls.join(head)+ls)
47             io1 = io.StringIO()
48             np.savetxt(io1,z[::-1])
49             vtk.write(io1.getvalue())
50             io1.close()
51             #
52             subhead=['SCALARS_%s_%s_1'%(basin,'int'),'LOOKUP_TABLE_default'

```

```
53     vtk.write(ls.join(subhead)+ls)
54     io1 = io.StringIO()
55     np.savetxt(io1, basin[:, :-1], fmt='%d', delimiter='_')
56     vtk.write(io1.getvalue())
57     #
58     subhead=['SCALARS_%s_%s_1'%(var, 'float'), 'LOOKUP_TABLE_default']
59     vtk.write(ls.join(subhead)+ls)
60     h=np.loadtxt(fn, skiprows=6, dtype=float)
61     io2 = io.StringIO()
62     np.savetxt(io2, h[:, :-1], fmt='%f', delimiter='_')
63     vtk.write(io2.getvalue())
64     #
65     vtk.close()
```

---