DECIPHeR

User Manual

Code Release Version 1

Gemma Coxon, Jim Freer, Rosie Lane, Toby Dunne

School of Geographical Sciences
University of Bristol, United Kingdom

August 2018

Authors:

Dr Gemma Coxon

Prof Jim Freer

Rosie Lane

Toby Dunne

School of Geographical Sciences

University of Bristol

United Kingdom

www.bris.ac.uk/geography

Email Contact : gemma.coxon@bristol.ac.uk

DECIPHeR Downloads : https://github.com/uob-hydrology/DECIPHeR

Disclaimer

Before using DECIPHeR please read carefully the following terms for use. Downloading the DECIPHeR model code (“the Software”) on to your computer indicates you accept the following terms:

1. The developers do not warrant that the software will meet your requirements or that the operation of the software will be uninterrupted or error-free or that all errors in the Software can be corrected.

2. You install and use the Software at your own risk and in no event will the developers be liable for any loss or damage of any kind including lost profits or any indirect incidental or other consequential loss arising from the use or inability to use the Software or from errors or deficiencies in it whether caused by negligence or otherwise.

3. The developers accept no responsibility for the accuracy of the results obtained from the use of the Software. In using the software you are expected to make final evaluation in the context of your own problems.

4. Users are not in reliance on any statements warranties or representations which may have been made by the developers or by anyone acting or purporting to act on their behalf.

Executive Summary

This document is the user manual for the open source implementation of the DECIPHeR hydrological model, version 1.0. It documents the code workflows, data requirements, input data formats and provides a walkthrough of all the codes.

DECIPHeR is a flexible modelling framework for hydrologic simulation and prediction from catchment to continental scales. The model can be adapted for specific hydrologic settings or data availability altering the representation of spatial heterogeneity, spatial connectivity and hydrological processes in the landscape. Crucially, the model has an automated build and is computationally efficient to run large ensembles and characterises model uncertainty.

There are two stages to the running DECIPHeR; firstly digital terrain analyses are performed to define the gauge network, set up the river network and routing, discretise the catchment into HRUs and characterise the spatial variability and hydrologic connectivity in the landscape, and secondly, HRUs are run in the rainfall runoff model to provide flow timeseries for locations on the river network specified by the user. Minimum data requirements include a digital elevation model and locations for streamflow output for the digital terrain analysis and rainfall and potential evapotranspiration to generate simulated discharge in the rainfall-runoff modelling.

DECIPHeR v1.0: Dynamic fluxEs and ConnectIvity for Predictions of HydRology

Overview

This document describes DECIPHeR (Dynamic fluxEs and ConnectIvity for Predictions of HydRology); a flexible hydrological modelling framework developed for research purposes by the University of Bristol.

DECIPHeR is designed for hydrologic simulation and prediction from catchment to continental scales. It provides the capability to tackle fundamental hydrological modelling challenges and address general as well as site specific problems. Importantly, more process complexity can be incorporated where needed to better suit local conditions (e.g. to account for human influences or more complex hydrological processes such as surface-groundwater exchanges) and thus account for spatial heterogeneities in hydrological response. Key features include:

1. Can be easily applied to single or multiple catchment(s), across large scales and complex river networks because it has an automated model building facility.
2. Can produce flow simulations for any gauged or ungauged point on a river network, and segment river reaches into any length for individual hillslope-river flux contributions.
3. Can (a) experiment with different landscape and climate attributes to characterise different spatial model structures and parameterisations and (b) modify the spatial scale/complexity of how spatial variability and hydrologic flow path connectivity are represented via its flexible modelling framework
4. Can easily allow different model hypotheses of catchment behaviour to be added because it is modular and extensible.
5. Can run large model ensembles to characterise model uncertainty because it is computationally efficient
6. Can be easily adopted and adapted because it is open source and includes a detailed user manual.

The model is described in detail in an open source, peer-reviewed paper in Geoscientific Model Development (Coxon et al, in review). Please cite this paper when you use the model.

In addition to this paper, we made the code open source on GitHub and wrote this user manual to allow other users to easily apply the model. This user manual is still a work in progress and new versions will appear over time as new modules are added, examples are updated and improved instructions are written. While every effort has been made to ensure the accuracy of the information contained within this user manual, the authors can not guarantee its accuracy and some errors/omissions may be present. We welcome advice from users to help improve the user manual.

Digital Terrain Analysis

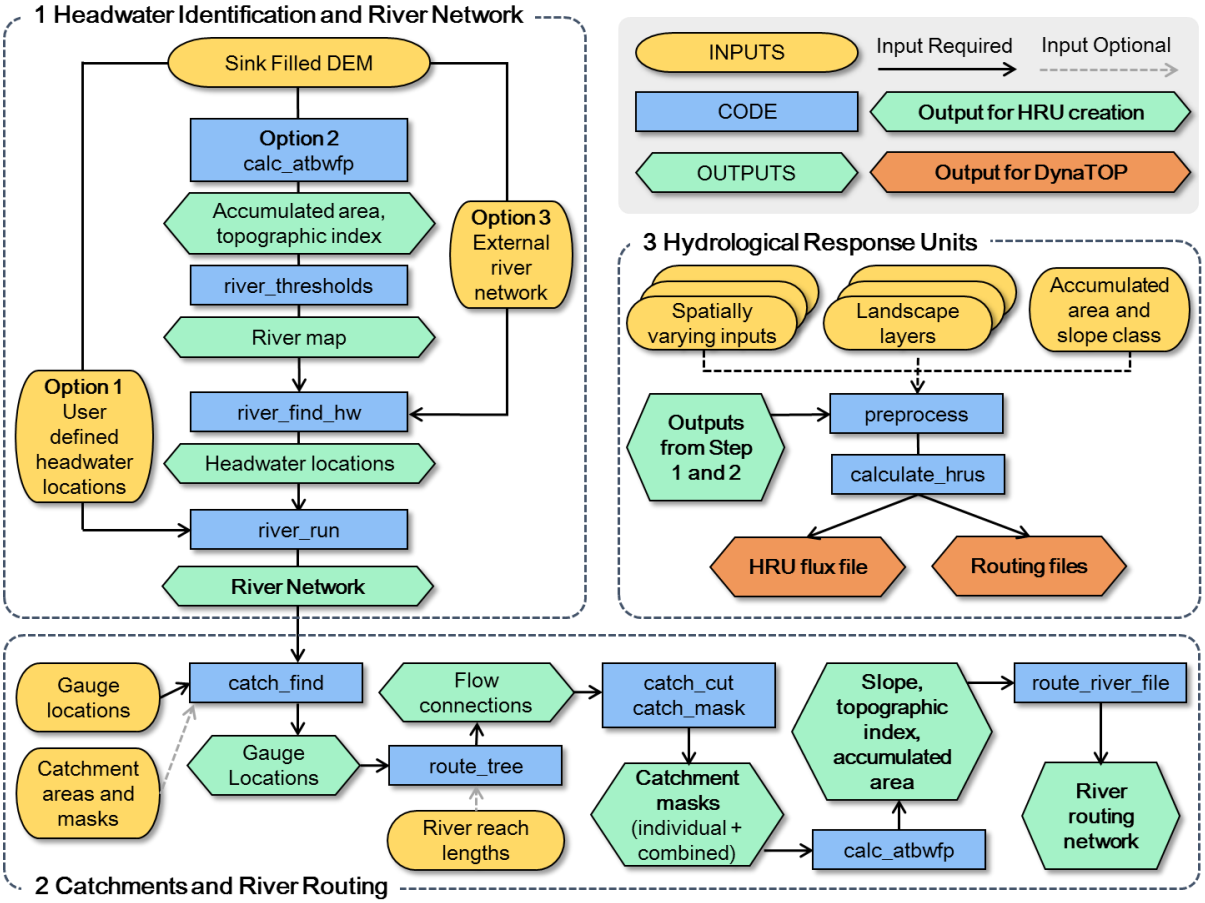
The digital terrain analyses in DECIPHeR are critical to define hydrological response units, characterise hydrological connectivity in the landscape, set up the river network and routing and determine where flows should be outputted either for gauged or ungauged locations. The codes take a digital elevation model and gauge locations supplied by the user to derive (1) a river network, (2) catchment masks, (3) routing tables, (4) slope, accumulated area and topographic index, (5) hydrological response units and flux files, for use in the hydrological modelling.

The process can be fully automated enabling multiple catchments or national scale data to be easily and quickly processed. There is flexibility for running the code depending on data availability (options for data sparse regions and including more information in data rich regions) and the ability to change default options depending on your requirements.

The code is FORTRAN 2003 compliant with new array and memory handling to allow any size dataset at any spatial scale and complexity of gauging network to be processed. Example settings files, including the gauge list files, HRU classifiers file and file specifying classes for slope, elevation and accumulated area are provided on github.

Code Workflow

The code workflow below shows the inputs required for the DTA, the code used and outputs produced.



Data Requirements for Digital Terrain Analysis

Data	Description	Essential?	Example file on github?
Digital Elevation Model	Digital elevation model in ascii format. Must contain no sinks or flat areas to ensure that the river network and catchments can be properly delineated.	Yes	No
Gauge List *Steps 0-2*	Gauge IDs and XY locations in DEM of the points on the river network where flow time series is needed (can be any gauged or ungauged point on the river network)	Yes	Yes
Gauge List *Step 3-4*	Gauge IDs of gauges you would like to calculate hydrological response units	Yes	Yes
Reference river network	River mask file in ascii format where any value >0 is classified as a river cell. Must be the same extent and spatial resolution as the DEM.	No	No
Reference catchment areas and masks	Catchment areas in km^2 can be supplied in the 'Gauge List' and catchment masks in ascii format can be used to select the best station location on the river network.	No	No
HRU Classifiers	File that specifies which classifiers to use to determine the hydrological response units	Yes	Yes
Classes for area, slope and elevation	Text file specifying classes for area and/or slope and/or elevation to split up the hydrological response units. Fractions must total 1.	No	Yes
Spatially Varying Inputs	Grids in ascii format specifying the ID for each input (rainfall and potential evapotranspiration). Must be the same resolution as the DEM.	No	No
Parameter or Model Structure Layers	Grids in ascii format specifying an ID for a parameter or model structure type. Must be the same resolution as the DEM.	No	No

Input File Formats for Digital Terrain Analysis

1 Digital Elevation Model

This file specifies the Digital Elevation Model used by the model and consists of a 2D raster array of ground elevations in ARC ascii raster format. The file consists of a 6 line header followed by the numerical values of each data point on the grid as a 2D array of i rows and j columns. The header lines detail the number of columns and rows, the x and y co-ordinate of the lower left corner of the grid, cell size and null value.

It is important that the DEM is sink filled and has no flats otherwise this will produce errors in the output from the DTA. It is also important that the DEM is in a projected co-ordinate system and the xllcorner/yllcorner and cellsize are provided in metres and not decimal degrees.

2 Gauge List (Steps 0 – 2)

This file specifies the list of gauge IDs and XY locations in the DEM of the points on the river network where flow time series is needed. These points can be any gauged or ungauged point on the river network. An example of this file is provided on github.

For each point where flow time series output is required, the format of the file is as follows:

Line 1: Comment line ignored by DECIPHeR

Line 2 -> n: Gauge ID, XY location of the gauge in the DEM, reference catchment area (km^2)

For example:

```
#GAUGE_ID     X       Y       CATCH_AREA
1       392436 830912 450
2       410113 848516 325
3       394684 830370 523
4       326202 954915 0
```

The gauge ID can be any number specified by the user but must be an integer. If the reference catchment area is unknown (for example, for an ungauged location) this must be set to zero as shown for Gauge ID 4 in the example above.

3 Gauge List (Step 3-4)

This file specifies the list of gauge IDs you want to calculate HRUs for. The code processes the gauge ID specified and all the gauge IDs upstream of this point so it is not necessary to list all gaugesAn example of this file is provided on github.

The gauge IDs need to be gauges that have been successfully processed in the previous DTA steps and will have three zeros added to the end of the original gauge ID e.g.

Line 1: Comment line ignored by DECIPHeR

Line 2 -> n : Gauge ID

```
GAUGES
1000
2000
```

4 Reference River Network (optional)

If required, headwater cells can be found from a pre-defined reference river network. This river mask file needs to be in ascii format where any value greater than 0 is classified as a river cell. Must be the same extent and spatial resolution as the DEM.

5 Reference Catchment Masks (optional)

If required, catchment masks produced by DECIPHeR can be checked against reference catchment masks to determine the best location for a gauge on a particular river. Reference catchment masks need to be in ascii format with any value greater than zero identified as being part of the catchment. The masks need to be at the same resolution as the DEM.

6 HRU Classifiers

This file specifies which classifiers should be used to determine the hydrological response units and which files to find the classifying data in. It has options to classify HRUs by

spatially varying inputs, different parameters and model structures and classes of area, slope and accumulated area. An example of this file is provided on github.

The code ignores any line starting with a '!'. It then looks for the following key words:

- TOPO_CLASS - to specify topographic classifiers (i.e. classes of slope, accumulated area and elevation). This should point towards a class_inc.dat file (see description below).
- INPUT_RAIN_CLASS – rainfall grid file (see description below of the expected format of the rainfall grid).
- INPUT_PET_CLASS – PET grid file (see description below of the expected format of the rainfall grid).
- MODELSTRUCT_CLASS – model structure file that identifies different model structure types in different parts of the landscape (see description below for the expected format of the model structure grid).
- PARAM_CLASS – parameter file that identifies different parameter types in different parts of the landscape (see description below for the expected format of the parameter grid).

The order of the keywords is not important, however, the keywords have to be specified exactly as above and need to be followed by an '=' and the file location with no spaces. An example is shown below.

```
!----------------------------
! HRU CLASSIFIERS
!----------------------------
TOPO_CLASS=/data/DTA/classinc.dat
INPUT_RAIN_CLASS=/data/DTA/REF_GRIDS/input_grid_10km.asc
!INPUT_PET_CLASS=/data/DTA/REF_GRIDS/input_grid_10km.asc
!MODELSTRUCT_CLASS=/data/DTA/REF_GRIDS/modelstruct_grid_hydrogeo.asc
!PARAM_CLASS=/data/DTA/REF_GRIDS/param_grid_soils.asc
```

In this example, rainfall and topographic classifiers are going to be used to split up the landscape as the other classifiers are commented out.

7 Gridded Inputs (optional)

These optional .asc files should be included if you want to use gridded precipitation and PET inputs. The purpose of these files is to tell the model what precipitation/PET input each grid cell should receive.

They must be .asc files, covering the same area and the same resolution as your DEM, with the same 6 header lines (see example below). Each cell in the grid needs to be labelled with an ID value. Each ID value will correspond with a cell in the gridded precipitation/PET data. The cell values should range from 1 to the number of precipitation/PET grids and must be integer types. Values of 0 can be given for areas not included in the modelling domain.

NB. If the gridded precipitation and PET have the same spatial resolution, the same ID grid can be used.

8 Model Structure/Parameter Types (optional)

These optional .asc files should be included if you want to run different model structures or different parameter types or parameter ranges across your catchment. The purpose of these files is to tell the model where you want to apply each different model structure or model parameter setup.

They must be .asc files, covering the same area and the same resolution as your DEM, with the same 6 header lines. The cell values should range from 1 to the number of different model structure or model parameter representations you want to include in the model, and must be integer types. Values of 0 can be given for areas not included in the modelling domain, in the example below the 0 values are outside the catchment boundaries.

The full filepaths to these files must also be added to the HRU classifiers file to tell the DTA to include these files when defining HRUs.

9 Classes for slope, elevation and accumulated area (optional)

The class_inc.dat file specifies percentiles for slope, elevation and accumulated area that are used to split up the landscape. An example of this file is provided on github.

The format of the file is as follows:

Any line starting with a '!' is ignored by the code.

Keyword Number of Fractions
Fractions

Keyword must be 'AREA', 'SLOPE' and 'ELEV' for accumulated area, slope and elevation respectively. The order of the keywords does not matter. The fractions must be separated by tabs and add up to 1.

An example is shown below. In this case, area, slope and elevation are all going to be used to determine the hydrological response units. For area and slope there are three equal classes while for elevation there are five classes.

```
! classinc.dat specifies the classification fractions for slope and accumulated area, these must sum up to 1!
AREA    3
0.34    0.33    0.33
SLOPE   3
0.34    0.33    0.33
ELEV    5
0.1     0.1     0.2     0.3     0.3     |
```

Output File Formats for Digital Terrain Analysis

The DTA codes output many useful datasets including:

- Ascii files of slope, accumulated area and topographic index
- A river network including distance to outlet
- Locations of headwater cells
- Routing tables detailing connectivity between gauges
- Individual catchment masks and nested catchment masks

These outputs from Step 1 and Step 2 from the DTA codes are described in detail in the DTA Walkthrough, while the HRU Flux and Meta file are described in more detail below.

1 HRU Flux File (*_dyna_hru.dat)

This file is output after running calculate_hrus.f90. It characterises the hydrological connectivity in the landscape by detailing the sharing from HRUs either to itself, to other HRUs or to rivers.

The header lines detail:

1. The gauge that this HRU flux file is for

2. The total number of HRUs
3. The catchment area
4. The number of classifying layers used to determine this HRU setup
5. The number of gauges in the catchment
6. The number of slope/area/elevation classes
7. The number of different input grids used
8. The number of parameter classes
9. The number of model structure classes

Items 6-9 in the list only appear if they are specified as a classifier in hru_class.dat.

After the header lines, the HRU fluxes are specified. For each HRU from 1 to n, the following information is provided:

Line 1 - HRU ID and then how many HRUs this HRU is sharing to.

Line 2 to number of sharing HRUs - HRU ID, Proportion of flow being shared to this HRU, Cumulative sum of the flow proportions

Line n – The proportion of flow being shared to river cells

```
HRU 21 Sharing to 19 HRUs
5      .000025        .000025
9      .159036        .159062
16     .001223        .160285
17     .000980        .161265
21     .229666        .390930
30     .094420        .485350
49     .000580        .485930
51     .080846        .566776
53     .001622        .568398
63     .210036        .778435
66     .000142        .778577
73     .007877        .786453
93     .002296        .788750
97     .114126        .902875
99     .000379        .903255
113    .022849        .926103
118    .001374        .927477
119    .000756        .928233
120    .061063        .989296
RIVS   .010704       1.000000
```

In the example above, HRU ID 21 is sharing to 19 other HRUs. It is sharing (for example) 11.4% of it's lateral subsurface flow with HRU 97, 21% of its lateral subsurface flow with HRU 63 and 22.9% of the lateral subsurface flow is remaining within this HRU. 1.07% of the lateral subsurface flow is being transferred into a river. The cumulative sum should always equal 1.

After the HRU to HRU fluxes, the HRU to river fluxes are then detailed, specifying which river(s) each HRU shares to. These are defined as the proportion of flow being shared to each river of the 'RIVS' percentages defined in the HRU to HRU fluxes.

```
HRU 19 Sharing to 1 RIVS
51     1.000000       1.000000
HRU 20 Sharing to 0 RIVS
HRU 21 Sharing to 1 RIVS
24     1.000000       1.000000
```

In the example above, of the 1.07% of flow being shared from HRU 21, 100% of this flow is contributing to river number 24.

2 HRU Meta File (*_hru_meta.dat)

This file is also output after running calculate_hrus.f90. It details the metadata for each HRU.

The header lines specify which catchment the HRU meta file is for, how many HRUs have been calculated for this catchment and the number of columns.

The first three columns specify the HRU ID, the number of cells contained within each HRU and the mean topographic index value of that HRU. Following this, the header columns ending in '_ID' will contain a number from 1 -> n which is used by the model to determine the inputs, model structure and parameter ranges to use for that particular HRU. For the example HRU metadata file below, HRU 1 will use rainfall and PET inputs from the second column of rainfall and PET inputs in the input file, it will use model structure number four as specified in the model structure settings file and the first set of parameter ranges specified in the parameter settings file. The gauge, area and slope IDs are not used by the model but are provided for user information.

These IDs are often obtained from larger datasets so the following columns ending in _NUM denote the original class of the dataset that was input into calculate_hrus.f90 from the hru_classifiers.dat file. In the example below, the PET and rainfall number from the input grid is 1152.

```
! HRU Meta File for Gauge 54001000
HRUs    2457
NUM_COLS      15
!
HRU_ID  NUM_CELLS  MEAN_ATB  GAUGE_ID  AREA_ID  SLOPE_ID  RAIN_ID  PET_ID  MODELSTRUCT_ID  PARAM_ID  GAUGE_NUM  RAIN_NUM  PET_NUM  MODELSTRUCT_NUM  P
1       5       5.11920       2       1       3       2       2       4       1       54065000        1152    1152    1       1
2       332     5.55328       19      1       3       50      50      3       2       54095000        938     938     2       2
3       13      5.57677       22      1       3       32      32      1       2       54094000        1022    1022    3       2
4       590     5.59697       29      1       3       56      56      1       2       54081000        891     891     3       2
5       1       5.61600       37      1       3       61      61      1       3       54034000        899     899     3       3
6       38      5.63703       15      1       3       20      20      1       1       54066000        1064    1064    3       1
7       55      5.67024       2       1       3       4       4       2       1       54065000        1109    1109    4       1
8       178     5.69872       22      1       3       31      31      1       2       54094000        1021    1021    3       2
9       325     5.71221       22      1       3       41      41      1       2       54094000        980     980     3       2
10      28      5.71293       12      1       3       20      20      1       1       54061000        1064    1064    3       1
```

All columns after the GAUGE_ID are only output to the table if the user includes them in the HRU classifier file.

Rainfall Runoff Modelling

DECIPHeR simulates and produces streamflow estimates at any point on the river network defined by the user. The code is modular and extensible to easily allow different model hypotheses of catchment behaviour to be added and tested by incorporating different model structures/parameterisations in different parts of the landscape. Spatially varying inputs can also be incorporated so that each HRU receives different precipitation and/or potential evapotranspiration inputs.

The process can be fully automated enabling multiple catchments or catchment configurations to be easily run. The code is FORTRAN 2003 compliant with new array and memory handling to allow large arrays to be processed. Example settings and project files are provided on github.

Code Workflow

The code workflow below shows the inputs required for the rainfall-runoff modelling, the code used and outputs produced.

Data Requirements for Rainfall-Runoff Modelling

Data	Description	Essential?	Example file on github?
HRU files	Files outputted from the DTA. HRU flux file (*hru_dyna.dat) and HRU meta table (*hru_meta.dat) required for rainfall-runoff modelling.	Yes	No
Routing Files	Files outputted from the DTA. Table specifying gauge connectivity (*flow_conn.dat), distance and change in elevation (*flow_point.dat) and river cell information (*riv_data.dat) is required for the river routing	Yes	No
Input Files	Separate text files for rainfall, PET and observed discharge in m/ts^{-1}.	Yes	No
Parameter File	Parameter file specifying parameter ranges for each parameter type	Yes	Yes
Model Structure File	Model structure file specifying the model structure for each model type	Yes	Yes
Project File	Project file which specifies the directory, settings file and output folder for each catchment that can be run	Yes	Yes
Settings File	Settings file for each catchment specifying the parameter, model structure, HRU, routing and input files to use	Yes	Yes

Input File Formats for Rainfall-Runoff Modelling

1 HRU Files

For the rainfall-runoff modelling, DECIPHeR requires the HRU flux and meta file which are outputted from the digital terrain analyses. The HRU flux file contains the flow distribution matrix which specifies the fractional flux out of each HRU to (1) itself, (2) another HRU or (3) a river. The HRU meta file specifies the number of cells, mean topographic index value and then the IDs for different classifiers (gauge, area, slope, elevation, inputs, parameter type and model structure type). These are read in by DECIPHeR so the model knows the inputs/setup for each HRU and need to be in the 'INPUT' folder.

It is important that these HRU files match the model setup you would like to run. For example, if you would like to run spatially varying rainfall then preprocess and calculate_hrus need to be run with a rainfall grid specified in the HRU classifiers file. Similarly, if you would like to vary the parameter ranges for different HRUs then preprocess and calculate_hrus need to be run with a parameter grid specified in the HRU classifiers file.

2 Routing Files

DECIPHeR requires river routing information to allow streamflow estimates to be computed at any point on the river network. These are outputs from the digital terrain analysis with the following files required in the 'INPUT' folder:

- ***_flow_point.txt** - Lists each gauge, length of river and elevation
- ***_flow_conn.txt** - Lists the downstream gauge for each gauge and the distance and slope between the two
- ***_riv_data.txt** - Routing information needed for every river cell

3 Input Files

Three separate input files are needed for the discharge, precipitation and PET. These files specify the observed timeseries. The format of the files is as follows:

Line 1: indicates what data the file contains (discharge, precipitation or PET) and the timestep used (hourly, daily, etc), the units that the data is in, and the time period that it covers.

Line 2: indicates the most downstream gauge in the catchment area.

Line 3: where the data was taken from.

Line 4: specifies the timestep used, the number of timesteps, the number of ID columns, and the null data value.

Line 5: left blank. This is ignored by the model.

Line 6: column headers for the year, month, day, hour, and IDs.

The ID numbers for the flow gauges, rainfall and PET need to be the values used by the model, and these are found in the '_hru_meta.dat' file. These values are in the columns labelled 'GAUGE_ID', 'RAIN_ID', and 'PET_ID'. They correspond with the 'GAUGE_NUM', 'RAIN_NUM' and 'PET_NUM' in the same row. 'GAUGE_NUM' is the original gauging station name. 'RAIN_NUM' and 'PET_NUM' are the rain/PET ID values used in the ascii grid created by the user. The columns labelled with the model ID numbers in the input files need to have the corresponding discharge, precipitation or PET timeseries below. Three examples of the input files are given below:

Discharge input file:

```
NIGER daily discharge data (m/day) 01-Jan-1980 --> 31-Dec-2000.
Extracted for Niger at 1134700
Extracted from the GRDC.
24       7671   7        -9999  (timestep_hours, n_timesteps, n_catchments, nodata_value)

YYYY       MM       DD       HH       CatID_1          CatID_2          CatID_3          CatID_4
1980        1        1       24     0.0003949230     0.0003001394     0.0000528856     0.0003107915
1980        1        2       24     0.0003972460     0.0003001394     0.0000528856     0.0003079004
1980        1        3       24     0.0003925999     0.0003001394     0.0000528856     0.0002992271
1980        1        4       24     0.0003902768     0.0002930240     0.0000528002     0.0002956133
1980        1        5       24     0.0003833076     0.0002503318     0.0000522021     0.0002804351
1980        1        6       24     0.0003740153     0.0002432164     0.0000521167     0.0002695935
1980        1        7       24     0.0003600768     0.0002432164     0.0000520312     0.0002616431
1980        1        8       24     0.0003531076     0.0002445101     0.0000510914     0.0002522470
```

Precipitation input file:

```
NIGER daily rainfall data (m/day) 01-Jan-1980 --> 31-Dec-2000.
Extracted for Niger at 1134700
Extracted from MSWEP 0.1x0.1 degree daily global rainfall product
24       7671   156      -9999  (timestep_hours, n_timesteps, n_RainIDs, nodata_value)

YYYY       MM       DD       HH       RainID_1        RainID_2        RainID_3        RainID_4
1980        1        1       24     0.0000000000    0.0000000000    0.0000000000    0.0000000000
1980        1        2       24     0.0000000000    0.0000000000    0.0000000000    0.0000000000
1980        1        3       24     0.0000000000    0.0000000000    0.0000000000    0.0000000000
1980        1        4       24     0.0000000000    0.0000000000    0.0000000000    0.0000000000
1980        1        5       24     0.0000000000    0.0000000000    0.0000000000    0.0000000000
1980        1        6       24     0.0000000000    0.0000000000    0.0000000000    0.0000000000
1980        1        7       24     0.0000000000    0.0000000000    0.0000000000    0.0000000000
1980        1        8       24     0.0000000000    0.0000000000    0.0000000000    0.0000000000
```

PET input file:

The example give below is for gridded PET, but if lumped PET data is used, then there would only be one PET_ID.

```
NIGER daily PET data (m/day) 01-Jan-1980 --> 31-Dec-2000.
Extracted for Niger at 1134700
Extracted from GLEAM 0.25x0.25 degree gridded PET global product.
24       7671   156      -9999  (timestep_hours, n_timesteps, n_PETIDs, nodata_value)

YYYY       MM       DD       HH       PET_ID_1        PET_ID_2        PET_ID_3        PET_ID_4
1980        1        1       24     0.0032739260    0.0034870440    0.0033965980    0.0033257020
1980        1        2       24     0.0031427110    0.0034204950    0.0032690890    0.0031540520
1980        1        3       24     0.0031630080    0.0034084010    0.0032932880    0.0032031380
1980        1        4       24     0.0031037470    0.0033742750    0.0032623040    0.0031653170
1980        1        5       24     0.0031929980    0.0034398780    0.0033450420    0.0032604660
1980        1        6       24     0.0031829610    0.0034822560    0.0033850940    0.0032870530
1980        1        7       24     0.0031972620    0.0033469200    0.0032683050    0.0031931110
1980        1        8       24     0.0033059350    0.0033600510    0.0032915720    0.0032458540
```

4 Parameter File

DECIPHeR can be run using either default parameter values, or through Monte-Carlo sampling of parameters between set parameter bounds. The parameter file allows the user to specify ranges or default values for each parameter, as well as defining the number of Monte-Carlo simulations to run.

An example layout of the parameter file is given below along with an example on github. Any text following an ! is ignored by the model. The first 5 lines must follow the format:

1. The number of simulations/ number of Monte-Carlo sampled parameter sets to run. Must be an integer between 1 and 99,999,999.
2. Seeds for the random number generator, for the Monte-Carlo parameter sampling. Must be 2 integers separated by a tab. Simulations started with the same seeds and same parameter ranges will always produce the same parameter sets.
3. Parameters for the kinematic solution. Separated by a tab.
4. The number of parameter layers and parameter names, separated by a tab. The number of parameter layers in the model will be 1 unless a param_class.asc file was provided for the DTA (see DTA section 8). The number of parameter names for the model to read in must be equal to the number of parameter names following the param_layer column header on line 6.
5. Line 5 must be left blank as the model skips this line.

Starting on Line 7, a table is given to define parameter values or parameter value ranges in each parameter class/ parameter layer. The column headings are important, as the model only reads in columns with headings which are an exact match to a known parameter. The first column must always be param_layer, giving the ID of the parameter layer. The following column headings must each refer to a parameter, starting with the parameter name followed by either _def (this is the default parameter value), _ll (this is the lower limit of the parameter value for Monte-Carlo sampling), _ul (the upper limit of the parameter value). Upper and Lower limits must both be provided, and if a default value is given Monte Carlo sampling will

not be applied to that parameter. A table giving the default parameter names is given below, and can also be seen in the example parameter file provided below.

It is important that the table of parameter values or limits is tab delimited, and that the number of rows and columns is consistent with the values given in line 4.

	Keyword	Description
Recognised parameter names	szm lnto srmax srinit chv td smax	Form of the exponential decline in conductivity. Effective lateral saturated transmissivity. Maximum root zone storage. Initial root zone deficit. Channel routing velocity. Unsaturated time zone delay. Max effective deficit of subsurface saturated zone.
Recognised suffixes	_def _ll _ul	This is a default parameter value, Monte-Carlo sampling is not required for this parameter. This is the lower limit of the parameter value, and Monte Carlo sampling should be carried out. This is the Upper limit of the parameter value, and monte Carlo sampling should be carried out.

Example parameter file (an example of this file is also provided on github):

```
10          ! Number of simulations you want to run
5     2000  ! Seeds for this run
1     0.8   0.000000001   ! Kinematic solution Parameters
3     14    !number of parameter layers(rows/hrus), number of param names (columns-1)

param_layer  szm_ll szm_ul lnto_ll      lnto_ul      srmax_ll     srmax_ul     srinit_ll    srinit_ul chv_ll     chv_ul td_ll  td_ul  smax_ll
1     0.001  0.07   -7     5     0.005  0.15   0     0.01   250   4000   0.1   40     0.2    3
2     0.005  0.007  -7     -2    0.005  0.15   0     0.01   1000  4000   0.1   1      0.2    3
3     0.01   0.04   1      4     0.005  0.15   0     0.01   3000  4000   0.1   40     0.2    3
```

5 Model Structure File

The model structure file specifies the model structure that should be used for each HRU. This must correspond with the gridded model structure layer used in the DTA e.g. if three different model structures were specified in the DTA then three different model structures need to be specified in this file. If no model structure layer was specified then only one model structure needs to be specified in the file.

NB. Currently only one model structure is currently implemented in DECIPHeR. If you wish to add different model components then these will need to be added to the source code.

Line 1 - 2: Header Line (not read in by DECIPHeR)

Line 3: The number of model structure types and number of model components

Line 4: Model Component Name

Line 5: HRU Model Component ID

Line 6: HRU Model Component ID

Line n ... : HRU Model Component ID

	Keyword	Description	Model Options
Recognised Model Component Names	rootzone	Controls the root zone formulation in the model	1
	unsatzone	Controls the unsaturated zone formulation in the model	1

For information on the model structure currently implemented, see the GMD paper.

Example model structure file (an example of this file is also provided on github):

```
! MODEL STRUCTURES

1       2       ! Number of different model structure types (must relate to HRU Meta File) and number of model components
model_layer     rootzone        unsatzone
1       1       1
```

6 Project.dat

The project file is a list of all of your DECIPHeR projects, giving information on the location of the key settings files for these projects. Each project is usually a setup of the model for a different catchment, as it is possible to have different inputs, catchment setups, settings and parameter values within a project.

The project file must be saved in the same folder as the DECIPHeR executable file, and the name project.dat cannot be changed as it is hardcoded into DECIPHeR.

An example of the project.dat file is available on github. The structure of the project.dat file can be seen in the example below, in which 3 projects are defined. The first line is a title line, which will be ignored by the model. The second line must start with an integer, telling the model how many projects to read in. It is very important that this number is equal to the number of projects in the file.

The three projects are then given, and it is possible to have any number of projects in this file as long as this is reflected in line 2. Each project must be preceded by an empty line which is ignored by the model. Each project must then have 4 entries, the order of which cannot be changed. The entries must consist of a 13 character name/description (which is ignored by the model), a tab and then the string to be read in to DECIPHeR. For a detailed description of each entry see the table and example project file below:

Necessary information for each project:

Name	Description
project_name	A name to help you identify this project. It is recommended that this includes the catchment name or number.
project_dir	The full filepath to the directory containing the SETTINGS and INPUT folders for this project.
projectfiles	The full filepath to the specific project settings file for this project. This specific project settings file can then contain any number of different catchment layouts and input files.
project_out	The full filepath to the output folder and beginning of the output filename for this project.

Example project.dat file:

```
! DYMOND Example Project File
2             ! total number of project files

project_name: Catchment 1
project_dir : /data/MODEL/Catchment1/
projectfiles: SETTINGS/settings
project_out : OUTPUT/Catch1_

project_name: Catchment 2
project_dir : /data/MODEL/Catchment2/
projectfiles: SETTINGS/settings
project_out : OUTPUT/Catch2_
```

7 Settings File (e.g. severn_settings.dat)

The settings file contains information on the location of all input and setup data required for a specific DECIPHeR project. Within a settings file it is possible to include data for multiple different catchment setups (for example, defining HRUs using a uniform vs 10km rainfall input, or splitting the catchment using different numbers of slope and accumulation classes). It is also possible to include information for different input data files (for example, if you have extracted input data for different time periods, from different sources, or different spatial resolutions). When the model is run, the user will be given the option of which catchment setup and which input dataset they wish to use.

Each line in the settings file is formatted to first have a 13 character description, then a tab, then the string to be read into DECIPHeR. The 13 character is for the user's reference only, and is not read in by the model. Anything following the tab will be read in as a string by DECIPHeR. The order of lines is hardcoded into DECIPHeR, and therefore cannot be changed.

An example settings file has been given below and is also provided on github. The first 6 lines must take the following format:

1. Title line to be ignored by the model.
2. Blank line – will be ignored by the model.
3. Param_file: Full filepath to the parameter file.
4. Mstruct_file: Full filepath to the model structure information file.
5. N_cat_setups: Integer giving the number of catchment setups specified in this settings file. This must be given in 4 spaces, with zeros in any unused spaces.
6. N_flow_setups: The number of different flow setups given in this file. Again, this must be an integer filling 4 spaces, with any unused spaces filled with zeros.

Lines 7-9 will be ignored by the model. It will then read in each catchment setup. Each catchment setup must have the following 6 lines of information, and be followed by an empty line:

Name	Description
setup_name	A name that will help the user identify what is unique about this particular catchment setup.
HRU_filepath	Filepath from the project directory to the _dyna_hru.dat file created in the DTA.
HRU_metafile	Filepath from the project directory to the _hru_meta.dat file created in the DTA.
flow_conn	Filepath from the project directory to the _flow_conn.dat file created in the DTA.

flow_data	Filepath from the project directory to the _riv_data.dat file created in the DTA.
flow_point	Filepath from the project directory to the _flow_point.dat file created in the DTA.

The model will ignore the next 2 lines as title lines, and then begin reading the n input setups. Each input setup must have the following 4 lines of information, and be followed by an empty line:

Name	Description
setup_name	A name to help the user identify what is unique about this particular set of input data.
flow_file	Filepath from the project directory to the discharge data input file.
precip_file	Filepath from the project directory to the precipitation input file.
evap_file	Filepath from the project directory to the PET input file.

Example settings file:

```
! DYMOND Settings File

param_file  : SETTINGS/params.dat
mstruct_file: SETTINGS/model_structure.dat
n_cat_setups: 0003
n_flow_setps: 0003

 --- DATA LOCATIONS AND SETTINGS FOR ALL CATCHMENT SETUPS (must be n_cat_setups entries): ----------------------

setup_name  : HRUS SETUP 1 - 6 classes of acc area, gridded 10km input grid
HRU_filepath: INPUTS/Catch1_HRU1_dyna_hru.dat
HRU_metafile: INPUTS/Catch1_HRU1_hru_meta.dat
flow_conn   : INPUTS/Catch1_flow_conn.dat
riv_data    : INPUTS/Catch1_riv_data.dat
flow_point  : INPUTS/Catch1_flow_point.dat

setup_name  : HRUS SETUP 2 - 3 classes of acc area, gridded 10km input grid
HRU_filepath: INPUTS/Catch1_HRU2_dyna_hru.dat
HRU_metafile: INPUTS/Catch1_HRU2_hru_meta.dat
flow_conn   : INPUTS/Catch1_flow_conn.dat
riv_data    : INPUTS/Catch1_riv_data.dat
flow_point  : INPUTS/Catch1_flow_point.dat

setup_name  : HRUS SETUP 3 - 3 classes of acc area, lumped input grid
HRU_filepath: INPUTS/Catch1_HRU3_dyna_hru.dat
HRU_metafile: INPUTS/Catch1_HRU3_hru_meta.dat
flow_conn   : INPUTS/Catch1_flow_conn.dat
riv_data    : INPUTS/Catch1_riv_data.dat
flow_point  : INPUTS/Catch1_flow_point.dat

 --- DATA LOCATION AND SETTINGS FOR ALL INPUT SETUPS (must be n_flow_setups entries) : ----------------------------

setup_name  : 10km gridded rain and pet 2001 - 2012
flow_file   : INPUTS/Catch1_10k_obsq.dat
precip_file : INPUTS/Catch1_10k_rain.dat
evap_file   : INPUTS/Catch1_10k_PET.dat

setup_name  : lumped rain and pet 2001 - 2012
flow_file   : INPUTS/Catch1_lump_obsq.dat
precip_file : INPUTS/Catch1_lump_rain.dat
evap_file   : INPUTS/Catch1_lump_PET.dat

setup_name  : lumped rain and pet 1995 - 2012
flow_file   : INPUTS/Catch1_lump95_obsq.dat
precip_file : INPUTS/Catch1_lump95_rain.dat
evap_file   : INPUTS/Catch1_lump95_PET.dat
```

Output File Formats for Rainfall-Runoff Modelling

1 Simulated Flow files (*.flow)

Simulated flow time series are output for each gauge for the same time period and time resolution as the input data. These flow time series are output as a separate text file for each simulation and contain the flow time series for all gauges.

The beginning of the output file name is specified in project.dat and is followed by the model simulation number. Each column contains a flow time series for a different gauge. The order of the gauges is the same as flow_point.dat. For example, the flow time series for the

first gauge specified in *_flow_point.dat will be in the first column, the flow time series for the second gauge specified in *_flow_point.dat will be in the second column and so on.

Simulated flow time series is provided in m/ts.

2 Result Metadata Files (*.res)

Result metadata files are also output alongside every simulation (*.res). These files detail:

- The parameter values used for each parameter class and parameter
- Summary statistics for each river reach. These are given in mm as a total for the whole simulation. Statistics are given for each river reach – the order of the rivers are the same order as specified in *_flow_point.dat. Summary statistics include:
    - Total precipitation input per river reach for the simulation
    - Total potential evapotranspiration input per river reach for the simulation
    - Total evapotranspiration removed from the soil root zone per river reach for the whole simulation

DECIPHeR Walkthrough

The DECIPHeR walkthrough takes you through all the steps and codes to run the digital terrain analysis and rainfall-runoff modelling for DECIPHeR.

Bash scripts containing all commands are available on github. The first script (Run_DTA_Example_Script_1.sh) covers the catchment identification and river routing steps and can be done over a large domain. The second script (Run_DTA_Example_Script_2.sh) carries out the preprocess and identification of hydrological response units steps, which are done after selection of a particular gauge or set of gauges (or specified ungauged catchment).

For every catchment you wish to run in the modelling framework you need to complete the DTA steps first to gain the required input for the rainfall-runoff modelling.

Step 0: DTA Setup

0A: Create a folder for your input data and results of DTA.

0B: Make sure all the necessary data for the DTA analyses are within this data folder:

See the list of data in the Data Requirements for DTA section.

0C: Copy the source code onto the server you are working on and compile the code

NB: You will need to gfortran to compile the code and install makedepf90
NB: Ensure that the permissions for executable files allow you to run them. Permissions can be changed using chmod command (e.g. $ chmod 775 atb_wfp.e).

0D: Running the DTA codes

The commands found in red text in this walkthrough can be copied and pasted to the command line in your terminal window (recommended for the first run) for each step.

Alternatively, if you want to run all the DTA codes in one go, complete 0E and then run the bash script by typing the red text below into the command line.

```
./ Run_DTA_Example_Script_1.sh
```

0E: Open your bash script file for running DTA (e.g. Run_DTA_Example_Script_1.sh). You must modify MAIN_DIR and CODE_DIR to refer to your working directories and change the DEM, gauge list and river file names.

```
# Define path to MAIN_DIR
# This is your working folder which must contain all input files.
# THIS MUST BE CHANGED.
MAIN_DIR=/data/DECIPHeR/DTA

# Full directory to folder containing source code
# THIS MUST BE CHANGED.
CODE_DIR=/data/DECIPHeR/DTA/SOURCE_CODE

#necessary files:
#THESE MUST BE CHANGED TO REFER TO YOUR DEM AND GAUGE LIST
#Currently looking for a file called DEM.asc and gauge_list.txt
DEM=DEM
GAUGES=gauge_list.txt
```

```
#Optional files:
#THIS MUST BE CHANGED TO REFER TO YOUR RIVER NETWORK
#Currently looking for a file called RIVER.asc
RIV=RIVER
```

0F: Navigate to your MAIN_FOLDER.

```
# Change directory to main directory - this is where input and output
files are
cd ${MAIN_DIR}
echo MAIN_DIR = ${MAIN_DIR}
```

Step 1: River Network Codes

1A: Decide which river network option you want to run

In this walkthrough, we provide two options to create the river network:

1. Option 1 creates a river map using the DEM by calculating topographic index and accumulated areas across the river domain and then applying a threshold in these two variables to define a river map (very useful where no river map information is available)
2. Option 2 uses an external river network.

If following option 2, skip to 1D.

1B: Calculate topographic index, slope and accumulated area from the DEM

Run this with only -dem command option to produce the topographic index and area accumulations where accumulated area accumulates within river cells and across catchment boundaries (we use the -mask and -river command options later in the walkthrough).

```
${CODE_DIR}/atb_wfp.e -dem ${DEM}.asc
```

Command options for atb_wfp.e (*optional command options in italics*):

- -dem DEM input file
- *-mask* Input mask file grid cells labelled with catchment
- *-river* Input river file grid where >0 is river cell

Output for atb_wfp.e for the example above:

- DTA_atb_wfp.log Log file containing record of input files, processes and output files
- _area.asc Ascii file containing the accumulated area of each grid cell
- _atb.asc Ascii file containing the topographic index of each grid cell
- _mfd_slope.asc Ascii file containing the slope of each grid cell

1C: Use thresholds in topographic index and accumulated area to define river cells

Creates river layer specified by thresholds in topographic index and accumulated area (i.e. any cell with a topographic index or accumulated area greater than the thresholds specified is classed as a river cell). Thresholds are set to default values (14 and 1250000 for topographic index and accumulated area respectively) if not specified in the command options as shown in the example below:

```
${CODE_DIR}/river_thresholds.e -dem ${DEM}.asc -atb ${DEM}_atb.asc -area
${DEM}_area.asc

RIV=river_thresholds
```

But can be easily modified to different thresholds e.g.

```
${CODE_DIR}/river_thresholds.e -dem ${DEM}.asc -atb ${DEM}_atb.asc -area
${DEM}_area.asc -atb_thresh 15 -area_thresh 1200000

RIV=river_thresholds
```

Command options (*optional command options in italics*):

- -dem DEM input file
- -atb Ascii file containing the topographic index of each grid cell
- -area Ascii file containing the accumulated area of each grid cell
- *-atb_thresh* Threshold value of topographic index (default 14)
- *-area_thresh* Threshold value of accumulated area (default 1250000)
- *-out_riv* Name of river output file (default river_thresholds.asc)

Output:

- DTA_river_thresholds.log Log file containing record of input files, thresholds used, processes, output files
- river_thresholds.asc Ascii file where any cell >0 is classed as a river cell. Grid cells are given a value of zero when neither threshold is exceeded, one where the cell exceeded the area threshold, two where it exceeded the topographic index threshold and three where it exceeded both thresholds.

1D: Find headwaters from input river network file

Algorithm finds all headwater cells from an input river network file. The code goes through two stages; (1) finds all possible headwater cells in 'Pass 1' (there will be many candidates along the river as a result of clumps) and then (2) checks all headwater cells from 'Pass 1' and removes very small headwaters (-search) and also moves headwaters on ridges slightly downstream (-move_downstream) to avoid leaving headwaters on a ridge (where they have a higher likelihood of flowing in the wrong direction).

```
${CODE_DIR}/river_find_hw.e -dem ${DEM}.asc -river ${RIV}.asc
```

Command options (*optional command options in italics*):

- -river River ascii grid file (used as a mask where any value > 0 is river cell)
- -dem DEM used as sea mask where all river cells will be joined to sea
- -outlets Easting and northings of outlets – rivers will connect to these outlets
- *-search* Smallest headwater will be approximately half of the search distance. Distance is in metres and default value is 500m.
- *-move_downstream* Distance that headwater is moved downstream. Distance is in metres and default value is 100m.

NB. Either -dem or -outlets need to be specified

Output:

- DTA_river_find_hw.log Log file containing record of input files, thresholds used, processes and output files
- _dist.asc Raster layer showing distance to outlet
- _HW_search_500m_100m.txt Selected headwaters using thresholds specified in command options

1E: Generate river network from headwaters

The headwater locations generated in the previous step are then routed downstream in a single flow direction via the steepest slope until reaching a sea outlet, other river or edge of the DEM. This creates the river network where each unique river section is labelled with a unique river ID.

```
${CODE_DIR}/river_run.e -dem ${DEM}.asc -headwater ${RIV}_HW_500m_100m.txt
```

If the user has generated their own list of headwater cells, then these can also be inputted by changing the -headwater command option to a text file with headwater IDs and XY locations.

Command options (*optional command options in italics*):

- -dem DEM input file
- -headwater List of headwaters id, easting and northing

Output:

- DTA_river_run.log Log file containing record of input files, processes and output files
- _riv.asc River raster layer - each unique river section is labelled with a river ID

Step 2: Catchment Identification and River Routing

2A: Locate points on the river network where output is required

The user can specify XY locations where they require output to be produced by DECIPHeR in a gauge list. These can either be gauged or ungauged points on the river network and the user simply needs to specify a gauge ID (must be an integer), XY location in the input DEM and a reference catchment area in 'Gauge_list.txt'. If the reference catchment area is unknown (for example, for an ungauged location) this must be set to zero.

The XY locations specified by the user may not lie on the river network created in Step 1 or might not provide an appropriate catchment mask and so the code must choose an appropriate point on the river network for each gauge. To do this, the code searches in a given radius (default value of 500m or can be specified by the user in the command options) for all possible river cells and generates a catchment mask for each point. If no river cells are found within the initial search radius, then the code expands the search radius up to four times the initial radius to find candidate river cells. If a reference catchment mask/area is available, then a score is generated by calculating the difference between the generated mask/area and the reference mask/area. All possible river cells for a given gauge are outputted in *_station_candidate.txt. The best candidate river cell for each gauge is outputted in *_station_match.txt. This is determined either by choosing the point with the best fit to a reference catchment mask/area or if no reference catchment areas/masks are available then it picks the nearest point on the river network as the location of the gauge (if there is one or more nearest points on the river network then it picks the cell with the largest catchment area). If the XY location listed in the original gauge list lies outside of the DEM or has no river cells within the maximum search radius or produces catchment masks that go beyond the DEM then it will not be outputted in *_station_match.txt.

It is recommended that the user checks the log files and both results files to see which gauges were processed.

```
${CODE_DIR}/catch_find.e -dem ${DEM}.asc -river ${DEM}_riv.asc -stations
${GAUGES}
```

NB. If this code is re-run, it only processes 'new' gauges and skips the already processed catchments. If you want to re-run it for all your gauges then delete the _station_candidate.txt and _station_match.txt file.

Command options (*optional command options in italics*):

- -dem DEM ascii grid file
- -stations List of station points with gauge ID, XY location in DEM and catchment area. Catchment area of 0 means that the closest point on the river network is chosen.
- -river River ascii grid
- *-ref_catch* Location of reference catchment masks in ascii format
- *-search_radius* Search radius to find candidate river cells (default is 500m)

Output:

- DTA_catch_find.log Log file containing record of input files, command options specified, processes and output files.
    - Also includes list of gauge IDs from gauge list and whether the code was able to find a best candidate river cell and reason why not (either point doesn't lie on the DEM or no river cells close by).
- _station_candidate.txt List of all candidate river cells for each gauge ID
    - Gauge ID
    - XY location of candidate river cell
    - Type of gauge (will always be 2)
    - Reference catchment area of gauge (km2)
    - Calculated catchment area (km2)
    - Score (difference between reference and calculated area or mask)
    - Distance from XY location in the gauge list to candidate river cell
    - Whether the catchment mask lied within the catchment boundaries (1 or 0)
- _station_match.txt List of best candidate river cell for each gauge ID
    - Gauge ID
    - XY location of candidate river cell
    - Type of gauge (will always be 2)
    - Reference catchment area of gauge (km2)
    - Calculated catchment area (km2)
    - Score (percentage difference between reference and calculated area or mask)
    - Distance from XY location in the gauge list to candidate river cell
    - Whether the catchment mask lied within the catchment boundaries (should always be 0)
    - How the best candidate river cell was chosen (by reference area or mask, or no reference)

2B: Choose gauge locations you would like to process and link all gauge locations on the river network

At this point, you will have a list of the best possible locations on the river for each gauge ID you listed in your original gauge list (apart from those that did not lie within the DEM or had no river cells within the search radius). However, these 'best' gauge locations may not be appropriate and so you may wish to filter out the gauges with poor scores. A filter score of 20 (i.e. generated catchment area will be within +/-20% of reference catchment area) is given as a default but this can be easily changed in the command options. This filter score is ignored if there is no reference catchment area or mask. There is an additional filter to stop catchments being processed if they are less than a certain size (default is 1km^2).

It is recommended that the user carefully checks the log file to ensure the catchments they want to process have not been removed by the filters.

This piece of code also links all the gauge locations on the river network to start setting up the routing and provides the option to set a 'river reach length parameter' where output time

series can be specified at regular reach lengths between gauges making it useful for coupling to a flood model (not shown in walkthrough)

```
${CODE_DIR}/route_tree.e -dem ${DEM}.asc -river ${DEM}_riv.asc -points
${DEM}_station_match.txt
```

Command options (*optional command options in italics*):

- -dem DEM ascii grid file
- -river River ascii grid
- -points Points on the river mask ascii file
- *-filter_score* Filter points by score (default is 20)
- *-filter_area* Filter points by area (default is 1km^2)
- *-reach_length* Optional automatic reach length between gauges (m)

Output:

- DTA_route_tree.log Log file containing record of input files, command options specified, processes and output files.
    - Also includes list of gauge IDs that were removed from processing depending on the filter options specified in the command options
- _riv_dist.asc Ascii grid specifying the distance to river outlet
- _riv_id.asc Ascii grid specifying the river ID for each river cell
- _flow_point.txt Lists each gauge, length of river and elevation
- _flow_conn.txt Lists the downstream gauge for each gauge and the distance and slope between the two

2C: Generate a catchment mask for each gauge location

```
mkdir masks
${CODE_DIR}/catch_cut.e -dem ${DEM}.asc -points ${DEM}_flow_point.txt -out
masks
```

Command options (*optional command options in italics*):

- -dem DEM ascii grid file
- -points Points on the river mask ascii file
- -out Folder to put all the catchment masks in

Output (mask_info.txt and all catchment masks will be in your masks folder):

- DTA_catch_cut.log Log file containing record of input files, command options specified, processes and output files
- Mask_info.txt Information about each gauge area
- Gauge_*.asc Ascii grid either the mask for each catchment

2D: Produce a nested catchment mask where all catchment masks are combined together

```
${CODE_DIR}/catch_mask.e -base ${DEM}.asc -tree ${DEM}_flow_conn.txt -
mask_dirs masks
```

Command options (*optional command options in italics*):

- -base DEM ascii grid file
- -mask_dirs Directory to find catchment masks
- -tree The routing tree file

Output:

- DTA_catch_mask.log Log file containing record of input files, masks used and output
- *_mask.asc Nested catchment mask

2E. Check nested catchment mask against the river ID mask

The nested catchment mask is then checked against the river mask to ensure that all river IDs match catchment IDs (essential for the hydrological response units). Corrects the river file (saves as *_riv_id_check.asc) and creates a log of mismatches for inspection (see DTA_mask_check.log).

```
${CODE_DIR}/mask_check.e -mask ${DEM}_mask.asc -riv_id ${DEM}_riv_id.asc
```

Command options (*optional command options in italics*):

- -mask Nested catchment mask
- -riv_id Ascii grid specifying the river ID for each river cell

Output:

- DTA_mask_check.log Log file containing record of input files, any changed river cells and output
- _riv_id_check.asc Checked river ID ascii grid file where rivers match catchment mask

2F: Calculate topographic index and accumulated area that stop accumulating down rivers and don't flow across nested catchment boundaries

Run this with -dem, -mask and -river command option to produce the topographic index and area accumulations where accumulated area doesn't accumulate within river cells and across catchment boundaries.

```
${CODE_DIR}/atb_wfp.e -dem ${DEM}.asc -river ${DEM}_riv_id_check.asc -mask
${DEM}_mask.asc
```

Command options for atb_wfp.e (*optional command options in italics*):

- -dem DEM input file
- *-mask* Input mask file grid cells labelled with catchment
- *-river* Input river file grid where >0 is river cell

Output for atb_wfp.e for the example above:

- DTA_atb_wfp.log Log file containing record of input files, processes and output files
- _riv_mask_area.asc Ascii file containing the accumulated area of each grid cell
- _riv_mask_atb.asc Ascii file containing the topographic index of each grid cell
- _riv_mask_mfd_slope.asc Ascii file containing the slope of each grid cell

2G: Generate the final information for the river routing files

```
${CODE_DIR}/route_river_file.e -dem ${DEM}.asc -river
${DEM}_riv_id_check.asc
```

Command options for route_river_file.e (*optional command options in italics*):

- -dem DEM ascii grid file
- -river River ascii grid file grid where >0 is river cell

Output for route_river_file.e for the example above:

- DTA_route_river_file.log Log file containing record of input and output files
- _river_data.txt Routing information needed for every river cell

Step 3: HRU Setup

3A: Create a new folder in your own filespace for the HRU flux files and catchment data

3B: Make sure all necessary settings files for the HRU analyses are within this folder:

As a minimum this includes:

- A file detailing the HRU classifiers you wish to use to split up the landscape
- A file detailing the list of gauges you wish to process

Also see the list of data in the Data Requirements for DTA section.

3C: Ensure you have run Steps 1 and 2 and you are happy with the output produced by the codes

3D: Decide how you want to split up the landscape

This is the most important part of the Digital Terrain Analyses as it determines how the catchment is split up and whether you want different parts of the landscape to have different inputs, parameters and model structures. This can be altered in the HRU classifiers file with grids detailing the parameters, inputs and model structures you want in each part of the landscape.

3E: Running the HRU codes

The commands found in red text in this walkthrough can be copied and pasted to the command line in your terminal window (recommended for the first run) for each step.

Alternatively, if you want to run all the HRU codes in one go, complete 3F and then run the bash script by typing the red text below into the command line.

```
./ Run_DTA_Example_Script_2.sh
```

3F: Open your bash script file for running the HRU codes (e.g. Run_DTA_Example_Script_2.sh). You must modify ROOT_FN, CODE_DIR and OUTPUT_DIR to refer to your working directories.

```
# Full directory to folder containing source code
# THIS MUST BE CHANGED.
CODE_DIR=/data/DECIPHeR/DTA/SOURCE_CODE

# Your working folder for the first part of the DTA analysis and the root
file name for the DEM
# THIS MUST BE CHANGED
ROOT_FN=/data/DECIPHeR/DTA/DEM

#The output folder you want to put the results in
# THIS MUST BE CHANGED
OUTPUT_DIR=/data/DECIPHeR/DTA/HRU

#Text file listing NRFA Gauge IDs that you want to model
GAUGELIST=${OUTPUT_DIR}/gauge_list.txt

#HRU Class file listing which classifiers you want to use to split up the
catchment
HRU_CLASS_FILE=${OUTPUT_DIR}/hru_class.dat

#Catchment mask file for this gauge ID (full filepath)
# THIS MUST BE CHANGED
CATCHMASK_DIR=/data/DECIPHeR/DTA/masks/
```

Step 4: Calculate HRUs

4A: Subset larger datasets to gauges of interest

This code subsets the larger datasets produced in Steps 1 and 2 for the individual gauges that you are interested in. The DEM, slope, accumulated area, rivers and gauge mask are always subset as are the three routing files needed to run the rainfall runoff model. Other layers are only subset if specified as an additional classifying layer in hru_class.dat.

```
${CODE_DIR}./preprocess.e -gaugelist ${GAUGELIST} -hru_class_file
${HRU_FILE} -root_fn ${ROOT_FN} -output_folder ${OUTPUTFOLDER} -
catchmask_folder ${CATCH_FN}
```

Command options for preprocess.e (*optional command options in italics*):

- -gaugelist List of gauges you want to process
- -hru_class_file HRU classifiers file
- -root_fn Root filename from the DTA to read in area, slope, masks etc.
- -output_folder Folder where you want the output data to be written to
- -catchmask_folder Folder that contains catchment masks

Output for preprocess.e for the example above (*only outputted if included in hru_class.dat*):

- DTA_preprocess.log Log file containing record of input and output files
- _dem.asc Ascii file with the elevation of each grid cell
- _slope.asc Ascii file with the slope of each grid cell
- _area.asc Ascii file with the accumulated area of each grid cell
- _riv.asc Ascii file with the river ID of each grid cell
- _mask.asc Ascii file with the gauge ID of each grid cell
- _flow_point.dat Lists each gauge, length of river and elevation
- _flow_conn.dat Lists the downstream gauge for each gauge and the distance and slope between the two
- _riv_data.dat Routing information needed for every river cell
- *_rain.asc Rainfall Grid*
- *_pet.asc Potential Evapotranspiration Grid*
- *_modelstruct.asc Model Structure Grid*
- *_param.asc Parameter Grid*

4B: Determine HRUs and calculate fluxes between HRUs

This code is the final step of the digital terrain analyses. It takes the subset gauge data created by preprocess.f90 and determines the HRUs as specified by the classifying layers in hru_class.dat. It also creates the flux file necessary for the rainfall runoff modelling.

```
${CODE_DIR}./calculate_hrus.e -gaugelist ${GAUGELIST} -hru_class_file
${HRU_FILE} -output_folder ${OUTPUTFOLDER}
```

Command options for preprocess.e (*optional command options in italics*):

- -gaugelist List of gauges you want to process
- -hru_class_file HRU classifiers file
- -output_folder Folder that contains the subsetted gauge data and where you want the output to be written to

Output for preprocess.e for the example above (*only outputted if included in hru_class.dat*):

- DTA_calculate_hrus.log Log file containing record of input and output files
- _hru_dyna.dat HRU flux file
- _hru_meta.dat HRU metadata file
- _hru_array.asc Ascii file with the HRU number of each grid cell for your chosen catchment

Step 5: Rainfall Runoff Modelling Setup

5A: Create folders for your rainfall runoff modelling

Below is a suggested folder structure that the rest of the walkthrough will refer to, however, other folder structures could be implemented.

- A main directory that holds project.dat
- Separate folders within your main directory for each catchment you wish to run
- An INPUTS, SETTINGS and OUTPUTS folder in each catchment folder

5B: Copy all the necessary data from the DTA analyses into the INPUTS folder for each catchment

This should include:

- Routing files
  - *_riv_data.dat
  - *_flow_conn.dat
  - *_flow_point.dat
- HRU files
  - *_dyna_hru.dat
  - *_hru_meta.dat

5C: Create input files and then copy into the INPUTS folder for each catchment

See 'Input File Formats for Rainfall-Runoff Modelling' for a description of the format of these files.

NB: Ensure that these input files match the information from the HRU files – for example, if you would like to run spatially varying rainfall then preprocess and calculate_hrus need to be run with a rainfall grid specified in the HRU classifiers file.

5D: Create settings files and then copy into the SETTINGS folder for each catchment

This should include:

- Settings File
- Parameter File
- Model Structure File

See 'Input File Formats for Rainfall-Runoff Modelling' for a description of the format of these files and/or examples on github

NB: Ensure that these settings files match the information from the HRU files – for example, if you would like to run two model structures then preprocess and calculate_hrus need to be run with a model structure grid specified in the HRU classifiers file.

5E: Create project.dat file and then copy into your main working directory

See 'Input File Formats for Rainfall-Runoff Modelling' for a description of the format of this files and/or example on github

5F: Copy the rainfall-runoff modelling source code onto the server you are working on and compile the code to create the model executable

NB: Ensure that the permissions for executable files allow you to run them. Permissions can be changed using chmod command (e.g. $ chmod 775 atb_wfp.e).

Step 6: Running a simulation

6A: Copy the model executable into your main directory that contains project.dat

6B: To run the model, open a UNIX/LINUX shell and at a command prompt type the name of the executable file generated by the compiler:

```
./DECIPHeR_V1.exe
```

6C: Choose your project file, HRU file and input file by entering the number of the file you wish to run and pressing enter

6D: The model should now be running and producing output in your OUTPUTS folder

Code Description Overview

Code	**Purpose**	**Command options** (optional commands in *italics*)	**Output**
Topographic index atb_wfp.e	Run this with only dem to produce the topographic index and area accumulations (area accumulates within river cells) these accumulations are used in * the lisflood coupling scenario. * river_thresholds Run with the DEM + the Catch Mask + river mask. This produces area accumulations that stop accumulating down rivers and also don't flow across nested catchment boundaries.	-dem *-mask* *-river*	DEM only DTA_atb_wfp.log _area.asc _atb.asc _mfd_slope.asc DEM+Catch+River Mask DTA_atb_wfp.log _riv_mask_area.asc _riv_mask_atb.asc _riv_mask_mfd_slope.asc
River from Thresholds river_thresholds.e [Optional, where no river file exists]	Create river based on thresholds. Default threshold values • atb = 14 • area = 1250000	-dem -atb -area *-atb_thresh* *-area_thresh* *-out_riv*	DTA_river_thresholds.log river_thresholds.asc
River Find Headwaters river_find_hw.e	Find headwaters on input river network file. (River file will be joined to the sea by single flow direction)	-river -dem -outlets *-search* *-move_downstream*	DTA_river_find_hw.log _dist.asc _HW_search_500m_100m.txt

Build River river_run.e	Build single flow direction river network from headwaters	-dem -headwater	DTA_river_run.log _riv.asc
Catchment Find catch_find.e	Find the best matching point on river network for each station point. Stations outside boundary are skipped as are catchment masks that go beyond the DEM.	-dem -stations -river *-ref_catch* *-search_radius*	DTA_catch_find.log _station_candidate.txt _station_match.txt
Route Tree route_tree.e	Link gauges and points on river network. Default threshold values • filter_score = 20 • filter_area = 1km^2	-dem -river -points *-filter_score* *-filter_area* *-reach_length*	DTA_route_tree.log _riv_dist.asc _riv_id.asc _flow_point.txt _flow_conn.txt
Catchment Cut catch_cut.e	Produce the mask files all points on river network.	-dem -points -out	DTA_catch_cut.log Mask_info.txt Gauge_*.asc
Catch Mask catch_mask.e	Combine the masks to create a single mask with the nested catchment masks.	-base -mask_dirs -tree	DTA_catch_mask.log *_mask.asc
Mask Check mask_check.e	Checks river cell id's against mask id's. Corrects the riv file (saves as xxx_checked.asc) Creates a log of mismatches for inspection.	-mask -riv_id	DTA_mask_check.log _riv_id_check.asc
Route River File route_river_file.e	Produce a list of cell information for every river cell, to be used by the routing algorithm. Used as input to dynamic topmodel.	-dem -river	DTA_river_file.log _route_river_file.txt