CHyF Tools: Flowpath Constructor and Catchment Delineator, v1.1

Canada Centre for Mapping and Earth Observation,
Centre canadien de cartographie et d'observation de la terre
Natural Resources Canada, Ressources naturelles Canada

Publication Date

2020 April 6

Copyright and Licence Notice

CHyF Website

https://github.com/NRCan/chyf

Authorship

Mark Sondheim
Canada Centre for Mapping and Earth Observation,
Natural Resources Canada
(mark.sondheim@canada.ca)
Emily Gouge, Chris Hodgson
Refractions Research, Inc.
(egouge@refractions.net, chodgson@refractions.net)

Series

This document is the second in the following series:

- *Volume 1: CHyF Conceptual Model*
- *Volume 2: CHyF Tools: Flowpath Constructor and Catchment Delineator*
- *Volume 3: CHyF Data for CHyF Services Implementation Specification*
- *Volume 4: CHyF Web Services and the CHyF API*

Table of Contents

1.0 Introduction

CHyF provides a series of preprocessing tools to simplify the use of CHyF services. The first two publicly available tools are the CHyF Flowpath Constructor and the CHyF Catchment Delineator.

Included in this document are the specifications for the input data for the tools as well as the output that they generate. It is assumed that the input data used for geoprocessing has already been converted to CHyF compliant hydrologic features, as defined by the input data schemas described here. Additional attributes may also be included; they will not be altered in any way.

Both tools have a command line interface that allows for various input parameters to be specified and both also make use of a properties file. The tools are designed to take advantage of a multi-threaded architecture, so significant speed increases are practical with the right computing environment. However, even on a suitable desktop computer, they are quite performant.

The software is open source under the Apache 2 license. It is written in Java SE 11 (Long Term Support). It can be run directly or by using Python scripts for example if desired.

Several screenshots showing inputs and outputs are shown in Appendix 4. Reviewing these first will give the reader a clear idea of what the software does.

2.0 Purpose

Currently, two preprocessing tools are available: the CHyF Flowpath Constructor and the CHyF Catchment Delineator. The intent of each is described below.

2.1 CHyF Flowpath Constructor

- The Flowpath Constructor may be used to generate all elementary flowpaths found in polygonal waterbodies, such as lakes, estuaries and double-line (i.e. comparatively wide) rivers, or subdivisions of such waterbodies. It may for example be of interest to break large lakes or rivers into multiple subdivisions.
- In some situations existing skeletons in lakes are available that are considered worth retaining, in terms of both their geometry and associated attributes. In this case, the Flowpath Constructor can be run such that only bank flowpaths are generated. Where a bank flowpath intersects an existing skeleton flowpath, the existing skeleton flowpath is broken into two skeleton flowpaths, with identical attribution.
- In addition to the skeletonizer the Flowpath Constructor includes a directionalizer that can determine the direction of a flowpath if newly created or if the direction is unknown.
- The tool can also assess whether a flowpath represents a primary or secondary flow; this is of particular interest in areas where the flowpaths do not all follow a dendritic pattern, i.e., at points of divergence. The resulting primary flowpaths do form a dendritic pattern. Any cycles in the network (where water can flow around a loop back to its starting position) are corrected where practical.

2.2 CHyF Catchment Delineator

- This tool is used to create CHyF compliant catchments from hydrographic data in combination with a gridded Digital Elevation Model that represents the elevation of the bare earth.
- It takes into account the surface hydrography and a digital elevation model representing the ground surface. In very flat areas where the elevation data may not be very helpful, the results approximate a medial axis approach, with each catchment boundary placed about halfway between the nearby linear or polygonal waterbodies. Where local elevation differences are more significant, the boundaries are located such that the slopes on either side are oriented toward different waterbodies. The software makes a natural and continuous progression between these cases.
- CHyF compliant catchments delineated by the tool include reach catchments and bank catchments. Water catchments, which may for example correspond to lakes, wide

streams, or estuaries - or subdivisions of such features - are provided by the user as input.

- The reach catchments, bank catchments and water catchments form a complete coverage of the Area of Interest.
- Empty catchments are out of scope with this version of the delineator. This means that depressions that do not contain linear or polygonal waterbodies and are not bordering such waterbodies will be absorbed into adjacent catchments.
- Special treatment of built-up areas is also out of scope at this time.

The tools are typically run sequentially with the Flowpath Constructor first, followed by the Catchment Delineator. In the material that follows, the discussion generally pertains to the two tools taken together, but clarity is provided where differences exist.

2.3 High Level Workflow

The primary sources for input data are assumed to be the National Hydro Network (NHN) or in the future the NHNv2 in Canada and the National Hydrography Dataset Plus (NHDPlus) or NHDPlus HR in the United States. However, regional datasets could also be used.

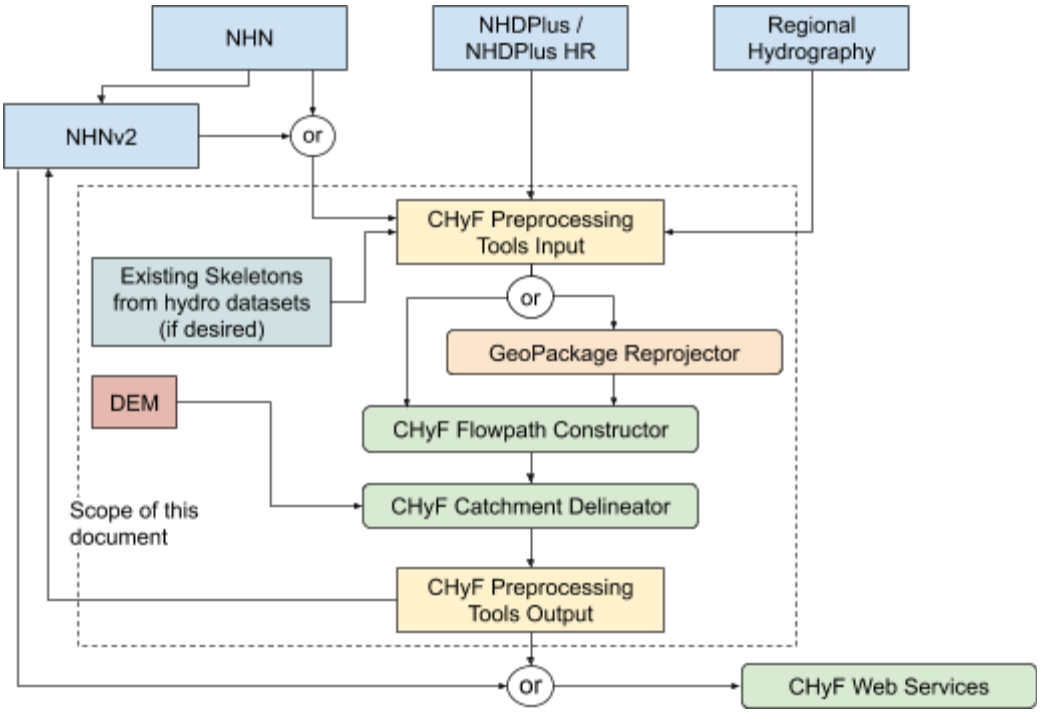


Figure 1: High level workflow

The CHyF Preprocessing Tools input includes reach flowpaths and water catchments, and optionally skeleton flowpaths, all as described in detail later in this document. The figure shows

the CHyF Flowpath Generator and the CHyF Catchment Delineator being used in sequence. This in fact is how they typically would be deployed, but using them independently of one another is also possible.

Figure 1 also shows that the GeoPackage Reprojector (Appendix 3, section 1) can be applied optionally. This simple command line utility allows the user to reproject the data, and to set a coordinate precision if desired, without needing to run a GIS or making use of other complex tools.

The CHyF Preprocessing Tools Output includes reach, bank and skeleton flowpaths, as well as reach, bank and water catchments. In the case of the NHN/NHNv2 these elements or a subset of them are copied into the NHNv2. The CHyF Preprocessing Tools Output can also be used as input to CHyF Web Services.

3.0 Data Extents and Coordinates

This section describes the extent of the data as related to one or more of the preprocessing applications. As it is assumed that both current applications will be run, it does not distinguish what is specific to each of them. Coordinate values of input and output data are also reviewed.

3.1 Data Extents

The diagram below shows the various data extents. These include the Area of Interest (green boundary with diagonal lines), the Processing Area (gray), and a coverage of DEM tiles (boxes). Descriptions are found below.

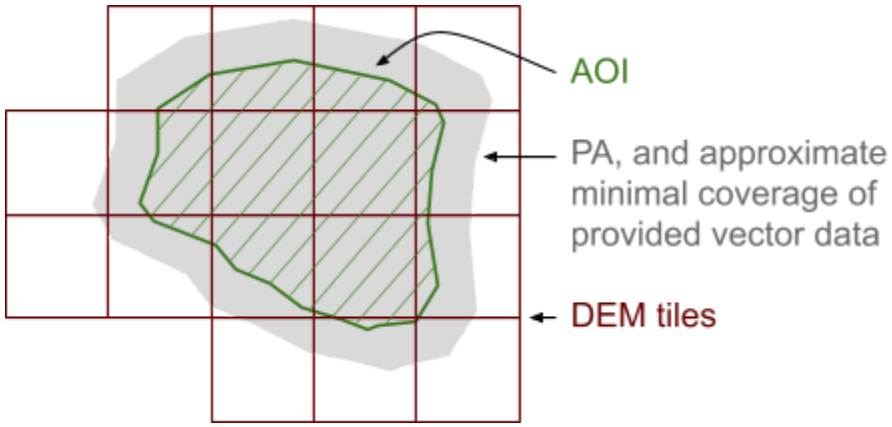


Figure 2: Data extent of AOI, Processing Area, and DEM

3.1.1 Area of Interest (AOI) – provided by the user

This is the area of interest represented as a simple polygon without holes. Typically the AOI corresponds to a major drainage basin or a major component of it. If the AOI is not known with a high degree of accuracy beforehand, the tools will still work. Note that the AOI is also provided as an output, with adjusted boundaries as described later, and with all output flowpath and catchment data contained within this AOI.

Constraints

- If the AOI intersects polygonal waterbodies, such as lakes, estuaries or wide rivers, it must be correctly noded along the common boundary, i.e., the vertices must be identical along the sections of the respective boundaries where they touch. This means that the boundary of the AOI and the boundary of the contained water catchment are coincident along the path across the water.
- Along the coast the AOI may extend into the ocean and thus may contain offshore islands.
- The boundary of the AOI ideally does not cross any reach flowpaths; instead, it winds its way around the headwater streams that are within the AOI. If the AOI crosses such

streams, or larger waterbodies, then all of such intersected features will be treated as within the AOI.

3.1.2 Processing Area (PA) – system generated

The Processing Area (PA) is the term applied to an area larger than the AOI; it is system generated. The processing steps carried out by the CHyF preprocessing tools are applied to the PA to eliminate any boundary effects within the AOI. The user provides input data for an area larger than the AOI, as described later, but does not provide a PA polygon. If the user provides input data only up to the AOI boundary, then processing artifacts and omissions may occur along and near the AOI boundary.

Constraints

- The input vector data extends beyond the AOI boundary in all directions.

3.1.3 DEM

The Digital Elevation Model data is provided by the user as a series of gridded GeoTIFF tiles.

Constraints

- GeoTIFF tiles must be provided that extend beyond the AOI. The tile coverage should extend at least to the PA boundary. That boundary is not given, but for practical purposes, it can be assumed to be about 5 km in all directions beyond the AOI, when the nominal scale of the input vector data is on the order of 1:20 000 to 1:25 000. (5 km is equal to one-half of the length of a side of the processing block, as described in section 4.5.)
- The grid spacing of the DEM should be approximately equal to (1/*hscale*) * 0.001, where *hscale* is the nominal data compilation scale of the hydrographic data, as represented by the original rivers and lakes data. For example, if the hydrographic data is considered to be at a scale of 1:20 000, then the grid spacing should be about 20 metres, although any spacing from roughly 10 m to 40 m should be acceptable. 2 m lidar derived DEM data should be resampled or decimated to create a 20 m grid. Higher resolution grids will require much longer processing times and the advantage of the higher density data is likely to be minimal, given the constraints imposed by the much coarser hydrography.
- The tiles must not overlap each other

3.2 Data Coordinates

The work carried out by the preprocessing routines occurs in 2D space. However, the incoming data may be in 2D (x,y), 3D (x,y,z), or 3D (x,y,m).

4.0 Input Data High Level Description

All vector data is provided as a single geopackage. The gridded elevation data is provided as a set of GeoTIFF tiles. The tools are each run through a Command Line Interface (CLI) that accepts a series of parameters. A properties file can be supplied to fine tune the processing carried out by each tool. If a properties file is not supplied, defaults will be used.

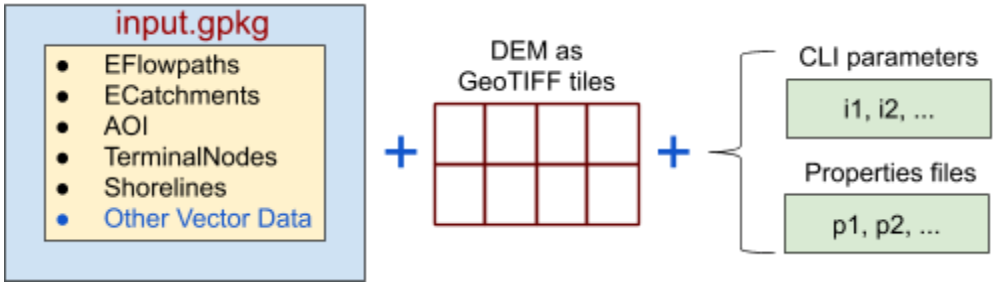


Figure 3: Input data, including a geopackage, geotiff tiles, and parameters

4.1 Packaging for Input Vector Data

- All vector data will be provided as a single GeoPackage containing a series of spatial tables (layers) as described in the table below. All the tables are required to have their SRID (spatial reference identifier) values specified correctly and they should all be the same SRID.

Spatial Table	Type Required In Table	Required
EFlowpaths	Flowpaths with ef_type=1 (reach flowpaths); ef_type=3 (skeleton flowpaths) optional	In all cases
ECatchments	Catchments with ec_type=4 (i.e., water catchments)	In all cases
AOI	--	In all cases
TerminalNodes	--	In all cases, where flow enters or exits the AOI; sink points within the interior of the AOI may or may not be provided
Shorelines	--	If the AOI intersects (or covers) all or part of a shoreline
(Other spatial tables)	--	No. If other tables are present, they will be ignored.

- Other spatial tables (layers) may be present, but they are not processed in any way by either the Flowpath Generator or the Catchment Delineator.

4.2 Packaging for Input DEM Data

All DEM data will be delivered as a series of GeoTIFF files residing in the same folder.

4.3 Projection for Input Data

- The user is responsible for providing the vector data in a suitable projection. A projection should be used that emphasizes the correct portrayal of shape, such as Lambert Conformal, although good results are also obtained with equal area projections.
- All vector layers provided must be in the same projection. All processing carried out by the CHyF Preprocessing Tools will be carried out in this projection.
- The DEM gridded data may be in the same projection as the vector data or in another projection. The projection information must be embedded in the GeoTIFF files. The GeoTIFF files may be in different projections, along an international border for example. In this case they must overlap to ensure that no gaps exist. The Catchment Delineator will make use of data in the overlap zone in an arbitrary manner, which should not matter if they were harmonized beforehand.

4.4 CLI Input Parameters and and Properties Files

Descriptions of the CLI input parameters and the properties file used for each tool are provided in Appendices 1 and 2. These descriptions are located within their respective initial sections, both entitled *Running the Software*.

4.5 Reserved Words

Reserved words are listed here, as related to the names of spatial tables in a geopackage and to the names of columns in a spatial table.

Restrictions	**Words**
Reserved words related to the names of spatial tables	
These words can only refer to the spatial tables as defined in section 4.1. If tables with any of these names are used for other purposes, the names should be altered.	EFlowpaths ECatchments AOI TerminalNodes Shorelines
Reserved words related to the names of columns in specific spatial tables	
EFlowpaths: within this table these column names must be used only as described in section 5.1.	ef_type direction_known name_string geometry
ECatchments: within this table these column names must be used only as described in section 5.2.	ec_type name_string geometry
AOI: within this table this column name must be used only as described in section 5.3.	geometry
TerminalNodes: within this table these column names must be used only as described in section 5.4.	flow_direction geometry
Shorelines: within this table this column name must be used only as described in section 5.5.	geometry

5.0 Input Data: Schemas

The names of all attributes (the names given under Column in the tables that follow) are case sensitive. The data must be provided with the correct column names.

5.1 Elementary Flowpaths

- The elementary flowpaths must be provided with an attribute named *ef_type*.
- All reach flowpaths must be included (ef_type=1). However, other types of elementary flowpaths may be provided, but they will not be used by the Catchment Delineator tool. Bank and optionally skeleton flowpaths are created by the Waterbody Skeletonizer tool.
- Other attributes include *direction_known* and *name_string*, as defined in the table below.
- The attribute *rank* is not provided on input, but it is on output. If it appears in the input data, the process defining rank will not run.
- Identifiers, names, length and other attributes may be given for each elementary flowpath provided, but they are not used by either of these tools.
- The attribute *name_string* will not be populated for any generated flowpaths. However, it will be practical to do so by running a separate tool. This tool (not yet developed) will also populate Strahler Order, Horton Order, Hack Order and length.

Feature: eflowpath (in spatial table: EFlowpaths)
Definition: an elementary flowpath conforming to the CHyF model
Attributes: ef_type, direction_known, name_string

Geometry: linestring

Constraints:

- The sequence of linestring vertices corresponds to the direction of flow if known.

Column	Required	Data Type	Description
ef_type	Yes	Integer	A code representing the type of elementary flowpath. (See the table below for valid values)
direction_known	No. If not supplied a value of -1 is assumed.	Integer	A code specifying whether the direction of the vertices corresponds to the direction of the flow. (See the table below for valid values)
name_string	No	String	The name, or a reference to a name, of the linear or polygonal waterbody containing the flowpath. The type is a character string, which may be a UUID, some other identifier, or a natural language name.

(other attributes)	No	--	Other attributes may be given, but will be ignored by CHyF tools and services.
geometry	Yes	LineString	Geometry representing the flowpath, with the sequence of vertices corresponding to the flow direction (if known).

ef_type		
Value	**Name**	**Description**
1	Reach	A reach flowpath corresponds to an observed or inferred single-line river. Includes *Observed* and *Constructed* from the NHN.
2	Bank	A bank flowpath connects a bank catchment to the flowpath network; it is otherwise analogous to a skeleton flowpath. This has no equivalent in the NHN.
3	Skeleton	A skeleton flowpath exists in a lake or double-line river; at both ends it connects to other skeleton flowpaths or to a reach flowpath. These are referred to as *Inferred* in the NHN.
4	Infrastructure	Reach flowpath representing a contained flow in a conduit, such as a storm drain, a sanitary sewer, a flow through a dam or an industrial complex. In the case of a flow through a dam, it is equivalent to *Constructed* for FLOW_QUALIFIER for a Water Linear Flow in the NHN.
direction_known		
Value	**Name**	**Description**
-1	Unknown	The direction of the flow is unknown and cannot be assumed to follow the order of the vertices. The CHyF Flowpath Constructor can be used to define flow direction; the order of the vertices may be reversed by the tool. The order may also be reversed if required to ensure that the skeleton flowpaths do not form a cycle.
1	Known	The direction of the flow is assumed to follow the order of the vertices. Either it was provided when input or it was assigned by the CHyF Flowpath Constructor.

5.2 Elementary Catchments

- The elementary catchments must be provided with an attribute named *ec_type*.
- All water catchments must be included (ec_type=4). No other catchment types are allowed as input.
- Identifiers, names, areas and other attributes may be given for each elementary catchment provided, but they are not used by either the Waterbody Skeletonizer or the Catchment Delineator.

Feature: ecatchment (in spatial table: ECatchments)
Definition: an elementary catchment conforming to the CHyF model
Attributes: ec_type, name_string
Geometry: polygon (may contain holes; multipolygon is not allowed)

Column	Required	Data Type	Description
ec_type	Yes	Integer	A code representing the type of elementary catchment. (See the table below for valid values)
name_string	No	String	The name, or a reference to a name, of the polygonal waterbody from which a water catchment was derived. The type is a character string, which may be a UUID, some other identifier, or a natural language name.
(other attributes)	No	--	Other attributes may be given, but will be ignored by CHyF tools and services.
geometry	Yes	Polygon	Geometry representing the area covered by a lake, an estuary, a double-line river, or a portion of any of these. A large lake may be subdivided into component pieces, each of which is treated as a water catchment. The same applies to estuaries and double-line rivers.

ec_type		
Value	**Name**	**Description**
1	Reach	An elementary catchment that contains a section of a single-line stream. The contained stream will bisect the catchment, except in the case of headwater stream segments or terminal stream segments.
2	Bank	An elementary catchment that is adjacent to a waterbody and that drains directly into it. It does not contain a waterbody. For example, if two streams drain into a lake, the remnant area between the catchments for the two streams also drains into the lake; it defines a bank catchment.
3	Empty	An elementary catchment consisting of internally drained land that does not touch a waterbody. In 2D the ring defining its boundary does not surround any waterbodies.
4	Water	An elementary catchment with polygonal geometry representing (i) watercourses such as rivers that are sufficiently large to have been mapped with polygonal geometry, or (ii) static or tidal waterbodies such as lakes, estuaries, nearshore zones, and the ocean.
5	Built-Up Area	Urban or industrial area, or area under construction. Such areas where constructed features may involve significant infrastructure developments that include conduits for the transfer of freshwater or waste water. Such conduits may cross one another and not intersect; as well, they may not drain the immediate area through which they pass.

5.3 Area Of Interest (AOI)

Feature: aoi (in spatial table: AOI)

Definition: The area of interest for which the respective preprocessing tools can be expected to provide accurate results

Attribute: This feature has no required attributes.

Geometry: polygon

Constraints:

- Multi-polygon not supported
- Simple polygon without holes

Column	Required	Data Type	Description
geometry	Yes	Polygon	This user supplied polygon defines the area of interest.

5.4 Terminal Nodes

Flowpath terminal nodes, represented as points, are typically located along the boundary of the AOI and represent inflows to the AOI or outflow from the AOI. What is required differs for water catchments (i.e., polygonal waterbodies or subdivisions of them) and reach flowpaths (corresponding to single-line streams).

For every location where a water catchment boundary is coincident with the boundary of the AOI, a midway point along the common bounding section (the intersection of the two boundaries) must exist in the TerminalNodes table that matches a vertex common to the two boundaries. This point must be identified as an inflow or an outflow.

By requiring both inflows and outflows, matching the results from one AOI to an adjacent AOI becomes practical operationally. If the implied direction calculated by the Flowpath Constructor at that point does not agree with the flow direction given at the terminal node, then the implied direction is treated as correct and the flowpath direction is considered to be in error.

For every location where a reach flowpath touches or crosses the boundary of the AOI, the intersection must be a point. If the direction of the flowpaths within the AOI in the general vicinity of that point are not known, then that point should exist in the TerminalNodes table, where it must be identified as an inflow or outflow.

Similarly, for each large, internally drained area that exists within the AOI, if the direction of flow is unknown for the associated flowpaths, then the point to be treated as at the bottom of the associated network is considered to be a sink. It should appear in the TerminalNodes table as an outflow. This will force all flows in that isolated network toward the sink point.

For isolated water catchments (lakes) and small networks without provided sink points, the tool will pick a sink point based on the direction of the flowpaths, if the flow directions are known. If the flow directions are not known, then where small streams connect to a lake, the lake is treated as receiving flow from the streams. If more than one lake exists or if no lakes exist, then the direction chosen through the small network will be arbitrary.

Feature: terminal_node (in spatial table: TerminalNodes)

Definition: A point on the boundary of the AOI where water enters or leaves the AOI, or a point acting as the sink of a large internally drained area if the directions of the flowpaths are not known.

Attribute: flow_direction

Geometry: point

Constraints:

- Must be located on an endpoint of a flowpath.
- Must be provided for any flowpath leaving the AOI if the flowpath direction is unknown.
- Where that flow is situated in a polygonal waterbody, it must also fall on a water catchment boundary vertex situated in the water and not on the shore on either side.

Column	Required	Data Type	Description
flow_direction	Yes (for points on AOI boundary)	Number	An indicator of the direction of flow, either into the AOI or out of the AOI.
geometry	Yes	Point	The point must be coincident with an endpoint of a flowpath.

flow_direction		
Value	**Name**	**Description**
1	in_flow	The water flows into the AOI.
2	out_flow	The water flows out of the AOI; for points acting as sinks of large internally drained areas, the lowest point in the associated network is designated as an out_flow.

5.5 Shorelines

- These must be provided only if the AOI touches or overlaps a shoreline.
- The shoreline features must form a continuous path without interruption (within the AOI) such that the boundary of the ocean is delimited everywhere, including where the ocean meets the land and where it meets freshwater. In this latter case, the shoreline must

match vertex for vertex where it meets a (fresh) water catchment. Where a reach flowpath enters the ocean, that flowpath endpoint must be coincident with a vertex on a shoreline feature.

Feature: shoreline (in spatial table: Shorelines)

Attribute: This feature has no required attributes.

Geometry: linestring

Constraints:

- Coastal islands may exist with polygonal geometry, but the shoreline in each case is a separate feature or series of features forming a counterclockwise ring.
- The delimiter between a double-line river or an estuary with the ocean must be defined as a shoreline feature.

Column	Required	Data Type	Description
geometry	Yes	LineString	The linestring is oriented, such that the coastal water is on the right side of the path defined from the first vertex to the last vertex on the linestring.

5.6 Other Geospatial Data

- Other layers of vector geospatial data may be included in the GeoPackage.
- No restrictions exist on the number of layers, their respective attributes, or their respective projections.
- The attributes of these layers, including their respective geometries, will not be altered in any way.

5.7 Digital Elevation Data

- DEM data will be provided as a folder of GeoTiff files with integer values as per the GeoTIFF specification.
- The cells in the DEM may or may not be square in the projection provided.
- The grid spacing should be as described in section 3.1.3.
- Conditioning of the DEM to ensure correct hydrologic behaviour is not required. The tools described in this report will work effectively in either case. In particular, catchment delineation should give good results unless the correspondence between the hydrography and the DEM is very poor, which generally is not the case.

6.0 Output Data: Packaging and Coordinates

The output data will consist of a geopackage containing equivalent spatial tables to those provided as part of the input geopackage, along with additional spatial tables that are used to store intermediate results during processing and can be useful for investigating concerns that arise from the output. These additional tables are identified in italicized green in figure 4 below.

The projection of all output data will be the same as the projection of the input data.

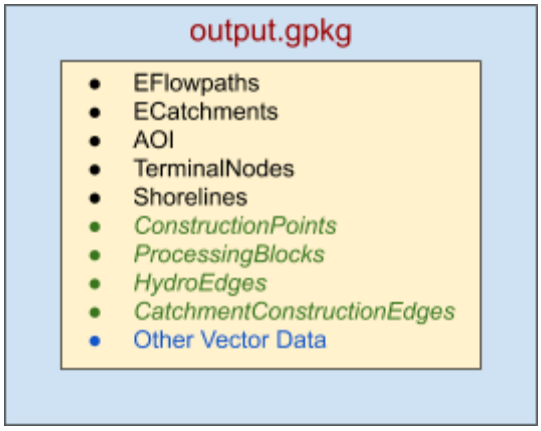


Figure 4: Output data, contained in a geopackage

7.0 Output Data: Schemas

The schemas for the five required input vector layers remain unchanged from the input data schemas described above (section 5.0). Additional clarification is provided below, as well as the schemas for the additional tables.

All UUID data types below are represented using the Text attribute type in the geopackage file. Geopackage does not support a uuid specific data type.

7.1 Elementary Flowpaths

- Elementary flowpaths in the output include reach flowpaths, bank skeleton flowpaths and bank flowpaths. The attribute *ef_type* is populated with a value of 1, 3, or 2, respectively, as defined earlier (section 5.1).
- Any additional attributes provided in the input eflowpaths will be included in the output.

Feature: eflowpath (in spatial table: EFlowpaths)

Definition: an elementary flowpath conforming to the CHyF model

Attributes: internal_id ef_type, name_string, rank, direction_known

Geometry: LineString

Constraints:

- The sequence of linestring vertices corresponds to the direction of flow if known or as assessed.

Column	Data Type	Description
internal_id	UUID	A generated universally unique identifier for the elementary flowpath.
ef_type	Integer	A code representing the type of elementary flowpath.
name_string	String	The name, or a reference to a name, of the linear or polygonal waterbody containing the flowpath. The type is a character string, which may be a UUID, some other identifier, or a natural language name.
rank	Integer	Rank is the term given to the distinction between a primary and secondary flow.
direction_known	Integer	This will always have a value of 1, unless the specific routine that determines the value for the attribute was not run.
(other attributes)	--	These are user supplied attributes, if any, that are carried through the process but not used. This is relevant only if the

		user has supplier flowpaths.
geometry	LineString	Geometry representing the flowpath, with the sequence of vertices corresponding to the flow direction.

The possible values for *ef_type* and *direction_known* are as listed in section 5.1 above.

rank		
Value	**Name**	**Description**
1	primary	Most flows are primary. However, at each point of divergence, where a flowpath separates into two (or more) flowpaths, one of the downstream flowpaths is designated as primary and the other(s) as secondary. In this case primary implies that the water volume is assumed on average to be greatest on the designated flowpath compared to the other flowpaths.
2	secondary	Immediately below a point of divergence, one or more flowpaths are designated as secondary if they do not carry the primary flow as specified above.

7.2 Elementary Catchments

- Elementary catchments with vector geometry are output from the Catchment Delineator tool. They form a continuous coverage for the area of interest.
- These elementary catchments have an attribute *ec_type* populated as *reach*, *bank* or *water* (as represented by a numeric code of 1, 2 or 4, respectively).
- Any additional attributes provided in the input ecatchments will be included in the output.

Feature: ecatchment (in spatial table: ECatchments) Definition: an elementary catchment conforming to the CHyF model Attributes: internal_id, ec_type, name_string Geometry: polygon (may contain holes; multipolygon is not allowed)		
Column	**Data Type**	**Description**
internal_id	UUID	A generated universally unique identifier for the elementary catchment.
ec_type	Integer	A code representing the type of elementary catchment. (See the table below for valid values). Codes for the

		various types are given in section 5.2 above.
name_string	String	The name, or a reference to a name, of the polygonal waterbody from which a water catchment was derived. The type is a character string, which may be a UUID, some other identifier, or a natural language name. If name_string was not provided on input, then it is not present in the output.
(other attributes)	--	If other attributes were provided on input, they are passed through.
geometry	Polygon	Geometry representing the area covered by a lake, an estuary, a double-line river, or a portion of any of these. A large lake may be subdivided into component pieces, each of which is treated as a water catchment. The same applies to estuaries and double-line rivers.

7.3 Area Of Interest (AOI)

- The AOI will have additional vertices added to the boundary where generated flowpaths intersect it. The area of the union of all catchments will not necessarily match exactly to the input AOI. It is left to the user to build a new AOI based on the generated catchments, if so desired.

7.4 Terminal Nodes

- No changes are made to the set of terminal nodes.

7.5 Shorelines

- Other than the possible inclusion of additional vertices where the shorelines are intersected by newly generated flowpaths, no changes are made.

7.6 Construction Points

- These points are calculated by the tool and provided as output. However, they may not be of interest to an end user, as they are not used as input into other tools.
- This table is available only in the output geopackage. There is no input geopackage equivalent.

Feature: constructionPoint (in spatial table: ConstructionPoints)

Definition: A construction point is a point along the boundary of a water catchment (equivalent to a lake, estuary, nearshore zone, or river, or a subdivision of such a feature) where flowpaths either enter or exit the polygon in question.

Attributes: internal_id, ecatchment_id, node_type, flow_direction

Geometry: point

Constraints:

- The sequence of linestring vertices corresponds to the direction of flow if known.

Column	Data Type	Description
internal_id	UUID	A generated unique identifier for the construction point.
ecatchment_id	UUID	The internal_id of the water catchment to which the construction points refers.
node_type	Integer	The type of node
flow_direction	Integer	A code representing the direction of flow at the point, with respect to the water catchment boundary on which it is located. The water catchment in question is identified by the value for ecatchment_id.
geometry	Point	The point is coincident with a vertex on the boundary of the water catchment and also coincident with the endpoints of inflowing and outflowing flowpaths.

flow_direction

Value	Name	Description
1	in_flow	The water flows into the polygonal waterbody.
2	out_flow	The water flows out of the polygonal waterbody.
3	unknown	The flow direction cannot be determined.

node_type		
Value	**Name**	**Description**
1	bank	A bank nexus is a point representing the interface between a bank catchment and a water catchment.
2	flowpath	A flowpath nexus is a point representing the interface between a reach catchment and either another reach catchment or a water catchment.
3	water	A water nexus is a point representing the interface between two water catchments.
4	headwater	A start point of a flowpath with no other flowpaths flowing into that location.
5	terminal	The endpoint on a flowpath with no other flowpaths flowing into that location.

7.7 Processing Blocks

- The processing blocks are created by the Catchment Delineator to divide the work into smaller pieces and track the status of the processing of each block.

Feature: processingBlock (in spatial table: ProcessingBlocks)

Definition: A processing block is a subdivision of the processing area into smaller square blocks to be processed individually

Attributes: id, state

Geometry: polygon

Column	Data Type	Description
id	Integer	A generated unique identifier for the processing block.
state	Integer	An integer which represents the state of the processing of the block.
geometry	Polygon	A square polygon representing the bound of the processing block.

state		
Value	**Name**	**Description**
-2	FAILEDQA	The block has failed QA.
-1	ERROR	An error occurred while processing the block.
0	DISABLED	The block has been disabled and will not be processed.
1	READY	The block is ready to be processed.
2	ASSIGNED	The block has been assigned to a processor.
3	EXTRACT	The data is being extracted for the block.
4	BUILD	The block is being processed.
5	QA	The block is being QA'd.
6	COMPLETE	The block is complete.

7.8 HydroEdges

- The hydro edges include reach flowpaths, shorelines, and the edges of water catchments. These edges have been broken into smaller pieces and assigned drainageIds based on which edges require catchments created for them.

Feature: hydroEdge (in spatial table: HydroEdges)

Definition: A hydro edge is an edge of a water feature which will be associated with a bank or reach catchment.

Attributes: id, state

Geometry: LineString

Column	Data Type	Description
drainageId	Integer	A generated identifier which is shared among edges which are part of a single catchment.
waterSide	Character	A single character which specifies which side of the edge has water on it. Single line streams have water on neither side, shorelines and water catchments have water on one side.
geometry	LineString	A linestring representing the edge.

waterSide		
Value	**Name**	**Description**
L	Left	The edge has water on its left side.
R	Right	The edge has water on its right side.
B	Both	The edge has water on both sides.
N	Neither	The edge has water on neither side.

7.9 Catchment Construction Edges

- The catchment construction edges are segments of eventual catchment boundaries.

Feature: catchmentConstructionEdges
(in spatial table: CatchmentConstructionEdges)

Definition: Catchment construction edges are segments of eventual catchment boundaries. They are output from the individual block processing, and used in the final merge stage to build the complete Catchments.

Attributes: id, state

Geometry: LineString

Column	Data Type	Description
leftDrainageId	Integer	The DrainageId of the HydroEdges into which the area to the left of this edge drains.
rightDrainageId	Integer	The DrainageId of the HydroEdges into which the area to the right of this edge drains.
geometry	LineString	A linestring representing the edge.

7.10 Other Geospatial Data

- Any additional data in the input geopackage, spatial or otherwise, will be included in the output geopackage, unchanged, with the exception of coordinate rounding to a common precision, which is applied to all vector layers in the geopackage.

8.0 Processing

8.1 Concurrent Design

Both the CHyF Flowpath Constructor and the CHyF Catchment Delineator are designed to leverage concurrent processing using a multi-core computer architecture where practical. In both cases, significant aspects of the processing can be handled concurrently, with the degree of concurrency dependent on the number of CPU cores available with sufficient memory. The impact of this design is most pronounced with the Catchment Delineator.

In the case of the Flowpath Constructor, each polygonal waterbody is processed independently, while other computation is carried out in a sequential fashion. The Catchment Delineator divides the data into a grid of blocks, each of which can be processed independently, utilizing multiple threads. The results from the blocks are later merged together, in a single-threaded process which is much faster compared to the in-block catchment delineation processing.

8.2 Timings

The Flowpath Generator and Catchment Delineator were tested on five NHN watersheds, chosen because of their diversity. All of the hydro features are from the NHN, the DEM files for the first four areas are from CDEM GeoTIFFs and the the DEM files for the “BIG” sample area were taken from the US Cloud Optimized GeoTIFFs.

The 5 sample areas were:

- KOTL: The Kootenay Lake area along the southern BC border is very mountainous, with some flat areas
- Richelieu: The Richelieu Valley along the southern QC border is mostly flat with many irrigation channels.
- NS: This area encompassing Halifax has moderate topography and is situated along the coastline.
- SAS: In Saskatchewan, this area is dominated by wetlands, with large areas containing no streams or lakes.
- BIG: The Rainy River area - one of the cross-border harmonized watersheds; it is one of the largest NHN watersheds by data volume. It is dominated by water.

The table below contains a breakdown of the numbers of features in each test area and the processing time taken for each area.

	Feature Counts							Processing Time (minutes)		
	EFlowpaths				ECatchments			Flowpaths		Catchments
Dataset	Reach	Bank	Skel.	Infra.	Reach	Bank	Water	cores=1	cores=4	cores=4
NS	825	1288	2644	2	825	1569	637	1	1	53
Richelieu	6735	1452	3363	10	6694	1442	842	9	5	93
BIG (Rainy)	15306	28545	73959	609	15303	28170	9857	142	116	124
SAS	2999	4236	8020	19	2980	4215	2673	14	10	497
KOTL	20585	2920	6637	0	20542	2901	1154	12	8	217

The processing was completed on a modest desktop computer consisting of an Intel Core i7-7700 CPU @ 3.6GHz with 16GB of 2.4Ghz DDR4 memory in a dual-channel configuration. In no case was more than 12GB of memory used by the processing; in most cases approximate 8GB was enough memory for the processing itself.

The catchment processing is 99% parallelizable and so the time to process an area scales with the number of cores in use, as long as the system has enough memory for the processes. The flowpath processing is only partially parallelizable, and so we show the results for both single core and with 4-way parallelism enabled, to give some measure of the parallel scalability of the process. An alternative approach to take advantage of multiple cores would be to run the flowpath processing in single-core mode but process multiple different areas at the same time; this should scale ideally with the number of cores/processes run.

It should be noted that the SAS dataset includes areas of sparse water features and required processing with different parameters than the other areas in order to get a good result in a reasonable time. Even with the parameter changes it still took much longer relative to the number of features in the area. In particular to process the SAS dataset the BLOCK_BUFFER_FACTOR was set to 2 and the input DEM resolution was reduced by half to reduce the memory requirement and the number of triangles in each block, since the effective processing area for each block was increased 2.25 times.

Infrastructure flowpaths were defined from NHN network linear flows that are coded as 3, "constructed". Because some are assumed to be buried pipes, they were ignored when the catchments were made. They could easily be recoded to be equivalent to reach catchments if desired; in this case the Catchment Delineator should be rerun. For test purposes here, this is not an issue.

9.0 Known Issues

9.1 Areas of Sparse Water Features

The Catchment Delineator divides the data to be processed into a grid of blocks; by default the blocks are 10 km square. In order to ensure consistent results across block boundaries, each block is buffered by default 50% of its width in all directions. In some areas where the water features are very sparse, this block size and buffer size are not enough to maintain consistency across the block boundaries.

One test area included an area where there was a circle 10 km in diameter that included no water features (flowpaths or water catchments). In order to process this area correctly, the block buffer had to be increased to 100% (property BLOCK_BUFFER_FACTOR = 1). The side effect of this increase to the block buffer is that the amount of DEM data being processed into a triangulation is more than doubled. This increased the memory required to process a block beyond what the test system had available, based on using 4 cores simultaneously; this would have necessitated reducing the test to using a single core if no other changes were made. Instead, and in order to process this in an acceptable time, it was necessary to resample the DEM data by 50%. This was done using gdal_translate (see Appendix 3 section 2).

The triangle-trickling process used to build the catchments is sensitive to areas where the distance to water is large, as the time to trickle an area is proportional to the square of the distance to water. Consequently the time to complete for this test area was much larger than occurred in other otherwise comparable test areas.

It may be possible in the future to adapt the process to dynamically adjust the block buffer based on the maximum distance to water within the block. However, the DEM data used to process all of the blocks must be the same, so the DEM cannot be dynamically resampled on a block-by-block basis.

9.2 GeoTIFF Compatibility Issues

The GeoTIFF DEM files must be readable using the GeoTools Java library, which internally uses the Java ImageIO library. An issue has been identified with DEM files available from the USGS (and possibly elsewhere) in which the 32-bit floating point values tiles are compressed using a predictor value which is not compatible with Java ImageIO. In these cases an error about an invalid predictor value will be output from the catchment delineator application. To resolve this, the data must be converted to a GeoTIFF which doesn't make use of a predictor for floating point compression. This can be done using gdal_translate (see Appendix 3 section 2).

Appendix 1: CHyF Flowpath Generator Details

1.0 Running the Software

To run the software use the .bat (windows) or .sh (linux) files provided as follows:

```
<tool>.bat [OPTIONS] <INFILE> <OUTFILE>
```

For example, to run the flowpath constructor on the sample data use the following command:

```
flowpath-constructor.bat ./testdata/Richelieu.32618.gpkg
./testdata/out.gpkg
```

To run the bank only constructor use the bank-only-constructor file:

```
bank-only-constructor.bat ./testdata/Richelieu.32618.gpkg
./testdata/out.gpkg
```

1.1 Options

The following options can be provided to both commands.

Option	Description
-p	Provides a custom property file.
-c	The number of cpu cores to use.

For example:

```
flowpath-constructor.bat -p custompropertiesfile.properts -c 14
./testdata/Richelieu.32618.gpkg ./testdata/out.gpkg
```

1.2 Logging

All logging is written to the console.
In addition, warnings and errors are written to a warnings.log file.

1.3 Advanced Options

As described above, these tools contain 5 separate processes:
1. Construction Point Generator

2. Skeletonizer
3. Directionalizer
4. Rank Generator
5. Bank Flowpath Generator

Each of these tools can be run independently as follows, modifying as required for your operating system. (These examples are for Windows. Linux uses : instead of ;)

1. Construction Point Generator

```
java -cp
lib/*;lib-chyf/chyf-core-1.0.0.jar;lib-chyf/chyf-flowpath-construct
or-1.0.0.jar
net.refractions.chyf.flowpathconstructor.skeletonizer.points.PointE
ngine [OPTIONS] <INFILE> <OUTFILE>
```

2. Skeletonizer

```
java -cp
lib/*;lib-chyf/chyf-core-1.0.0.jar;lib-chyf/chyf-flowpath-construct
or-1.0.0.jar
net.refractions.chyf.flowpathconstructor.skeletonizer.voronoi.Skele
tonEngine [OPTIONS] <INFILE> <OUTFILE>
```

3. Directionalizer

```
java -cp
lib/*;lib-chyf/chyf-core-1.0.0.jar;lib-chyf/chyf-flowpath-construct
or-1.0.0.jar
net.refractions.chyf.flowpathconstructor.directionalize.Directional
izeEngine [OPTIONS] <INFILE> <OUTFILE>
```

4. Rank Generator

```
java -cp
lib/*;lib-chyf/chyf-core-1.0.0.jar;lib-chyf/chyf-flowpath-construct
or-1.0.0.jar
net.refractions.chyf.flowpathconstructor.rank.RankEngine [OPTIONS]
<INFILE> <OUTFILE>
```

5. Bank Flowpath Generator

```
java -cp
lib/*;lib-chyf/chyf-core-1.0.0.jar;lib-chyf/chyf-flowpath-construct
```

```
or-1.0.0.jar
net.refractions.chyf.flowpathconstructor.skeletonizer.bank.Bank[OPT
IONS] <INFILE> <OUTFILE>
```

2.0 Software Components

The CHyF flowpath constructor consists of 5 distinct components:

- Construction Point Generator
- Skeletonizer
- Bank Flowpath Generator
- Directionalizer
- Rank Engine

2.1 Construction Point Generator

A2.2.1.1 Input Layers

EFlowpaths, ECatchments, AOI, TerminalNodes, Shorelines

The construction point generator creates a set of points that are provided to the skeletonizer as input/output points for polygonal waterbodies. For each waterbody, construction points are generated on the boundary of the waterbody polygon where water flows into or out of the waterbody. Bank construction points are also created at this time.

Construction points are created using the following rules:

- All eflowpaths that enter or exit a polygonal waterbody result in a construction point generated at the point of intersection with the waterbody on its boundary.
- Where two polygonal waterbodies touch along their respective boundaries, a construction point is created “near” the middle of the intersection. The property “vertex_distance” provided in the properties file determines what “near” means.

  First the center point of the intersection is computed. Then the closest vertex to this center point is found. If this vertex is within “vertex_distance” units from the center point, then the existing vertex is used as the construction point, otherwise a new vertex is added to both waterbodies at the center point and a construction point created at the center point.
- Where a waterbody intersects the coastline, the same process as above is used to generate a construction point for the intersection with the coastline.

- In cases where the waterbody intersects with the AOI, the software looks for a point provided in the TerminalNodes layer that lies on the intersection line (or point). If no TerminalNode point is found the software generates a warning and does not create a construction point here. Otherwise a construction point is created at the terminal node point.
- After all the above cases are processed, the number of construction points is added up. Each waterbody must have a minimum of two construction points. If zero construction points exist at this point (isolated lake), then two construction points are added at the start point on the boundary and halfway point around the waterbody. If a single construction point exists (headwater or terminal lake), then a single construction point is added at the point halfway around the waterbody from the existing construction point. When adding points halfway around the waterbody, existing vertices are used if they lie within the “vertex_distance” parameter from the midpoint, otherwise new vertices are added to the waterbody.
- Lastly bank construction points are generated. These are generated approximately halfway between all construction points generated above. However the following exceptions exist:
    - In cases where the waterbody intersects other waterbodies, the AOI, or the coastline, these intersection lines are excluded when computing halfway points.
    - Only a single bank construction point is generated for waterbodies with a single input or output point (headwater, terminal waterbodies).
    - As above vertices will be added if an existing vertex cannot be found that is within the “vertex_distance” from the computed point.

Implementation Note
Because processing one waterbody potentially affects other waterbodies (adding vertices along lines of intersection), this process is not parallelized.

2.2 Skeletonizer

The skeletonizer creates a connected network of eflowpaths among construction points associated with the waterbody.

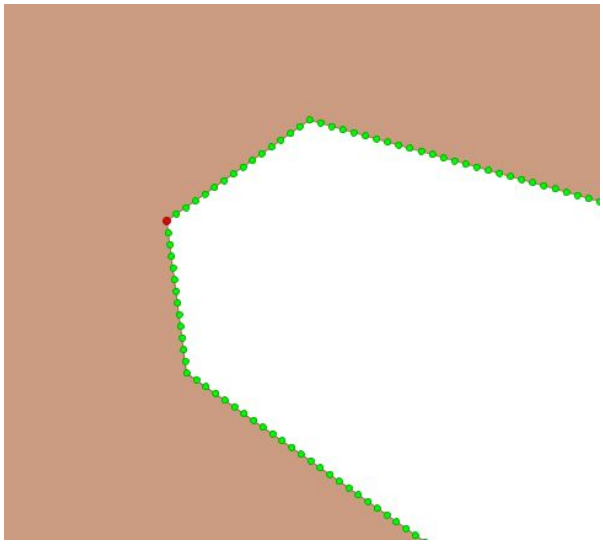
2.2.1 Input Layers

ECatchments, ConstructionPoints

Densification
The first step is to densify the waterbody boundary. A list of coordinates that make up the waterbody boundary is created. In cases where construction points lie on two adjacent coordinates, another coordinate in the midpoint of this segment is added to the collection. For

all coordinates that correspond to a construction point, that coordinate is removed and points are added on either side. The angle generated by the original coordinates determines how much densification is required. Acute angles require more densification. What defines an acute angle is specified by the “acute_angle” property in the properties file. At this point the coordinate array is used to densify the input points. The amount of densification required is specified by the “densify_factor” property in the properties file. Segments that require more densification are densified by 10 times this factor.

Example waterbody polygon



Zoomed in version where green dots represent the densified points.

A high densification value is required when waterbodies are narrow, otherwise the skeletonizer will not generate skeletons that remain inside the waterbody. However, there is a performance tradeoff; the software will run much slower with a high densification factor.

2.2.2 Voronoi

The set of densified points is passed into a Voronoi diagram builder. This generates a set of lineSegments (two point lines) which are clipped to include only segments that intersect the original waterbody. Then they are merged to form linestrings, and lines that do not touch a construction point or are necessary to connect the construction points are removed. Short linestrings are collapsed. Short is defined by the “minimum_skeleton_length” property. Lastly linestrings are simplified using the Douglas-Peucker algorithm with a factory defined by the “simplify_factor” property.

Voronoi line segments after the first of several trimming steps

2.2.3 Quality Assurance

A final QA process is run to make sure every construction point is connected to the skeleton network, and that no skeleton lines intersect the boundary of the polygon.

2.3 Bank Flowpath Generator

The bank flowpath generator can be used when skeletons already exist in the flow network and users want to continue to use these skeletons. In this case the bank flowpath generator will generate the bank flowpath skeleton lines that are a part of the CHyF model.

2.3.1 Input Layers

EFlowpaths, ECatchments, AOI, Shorelines

The bank flowpath generator uses the existing skeleton lines to break the waterbody into sub-polygons. For each part of these sub-polygons that corresponds to a boundary edge of the original waterbody a bank flowpath is added.

The first attempt to compute a given bank flowpath finds the nearest point between 20 and 80% along the boundary edge and existing skeleton lines. This percentage can be configured by the bank_node_distance_offset property. If set too small then bank flowpaths won't generate "nice" looking geometries; however, they will still be valid geometries. A straight line is made between these two points. If this straight line stays within the polygon and does not cross any other features it is used. Otherwise a skeletonizer process similar to what is described in the Skeletonizer section above is used to generate the bank flowpath for this polygon. In this case the input points are approximately halfway around the waterbody bank edge and halfway along the skeleton edges that make up the boundary of the polygon being processed.

All generated bank flowpaths are directionalized from the bank to the skeleton line.

2.4 Directionalizer

The directionalizer reads in the eflowpaths and directionalizes the network. The directionalizer attempts to respect the direction_known attribute on eflowpaths, if provided. If not provided it assumes everything is undirectionalized. In cases where it cannot respect the direction_known attribute, warnings are generated.

2.4.1 Input Layers

EFlowpaths, ConstructionPoints, Shorelines, TerminalNodes

The directionalizer algorithm is as follows:

1. Load all eflowpaths except bank flowpaths and build a graph.
2. Identify Sink Nodes
   Sink nodes are identified as the following (in this order):
   - All points from the TerminalNodes layer with a flow_direction of out_flow
   - All points where an eflowpath feature intersects a shoreline feature
   - All nodes in the graph where all touching eflowpath edges have a known direction and all flow into the node
   - All Construction Points with a FlowDirection of output
3. Directionalized bank flowpaths.
   These are directionalized so that the edge flows into a node in the graph created in the first step.
4. Directionalize from Sink Nodes.
   For each sink node:
   a. Compute a subgraph of all the edges that can be reached from this node (regardless of direction).

b. Find all sink nodes that are a part of this subgraph and remove them from the processing list.

c. Find all bridge edges in this subgraph.

d. Collapse all connected non-bridge edges into a single node. This will result in a tree data structure.

e. Directionalize the tree by walking up from each sink node (sink nodes are processed in order). Edge direction is respected here if known and can be retained if it does not create a cycle.

f. For each collapsed subgraph from step d:

    i. Compute the sinks and sources to this subgraph based on the directions assigned in step e.

    ii. Sort the sources based on distance to any sink (largest first).

    iii. For each source,
    Find the shortest path from the source to any sink. Keep an ordered list of these paths

    iv. Find an unvisited edge by iterating through the paths, walking down from the source to sink and finding an edge that has not been visited respecting direction if provided

    v. Find the “straightest” path (using angles as discussed elsewhere) from this edge to any sink.

        - If a straightest path cannot be computed, then compute the shortest path to any sink.
        - Trim the path, removing any edges that have already been directionalized.
        - Check if adding this path causes cycles, if it does flip the path.
        - If the path does not cause a cycle and the direction of all edges are unknown, then look at the angles generated at either end of the path. If the difference between these angles is greater than the value defined by the property “dir_angle_diff”, and the angles would be “better” if the path was flipped then try flipping the path. If a cycle is created then revert back the path. Otherwise keep the flipped path.
        A “better” angle means that the angle where the flow bifurcates is straighter. So if the start of the path forms an angle of 45, and the angle formed where the path joins back is 135, then this path would be flipped.
        - Add this newly created path to the front of the path list.

    vi. Repeat steps iv & v until there are no more edges to visit

5. At this point any non-visited edges form a subgraph that does not connect to any sink. These are likely small isolated lakes or small isolated connected segments. We compute a sink for these subgraphs and repeat step 4 for each subgraph.
6. Run a final cycle checked for validation.

2.5 Rank Engine

The rank engine determines which outflow is considered the primary outflow (main source of water flow) at each bifurcation of the flow network.

2.5.1 Input Layers

EFlowpaths, ECatchments, Shorelines

At each location where the flowpath bifurcates, the primary flowpath is computed based on angle of the flowpath and the channel width (in polygonal waterbodies).

In cases where channel widths cannot be computed for each output flowpath only angle is used to determine rank. Such cases include all single line flowpath cases and a few skeleton flowpath cases where all output paths do not reside in a distinct channel. In these cases the flowpath that forms the straightest angle is considered the primary flowpath. All other flowpaths are secondary. See below for details on how angle is computed.

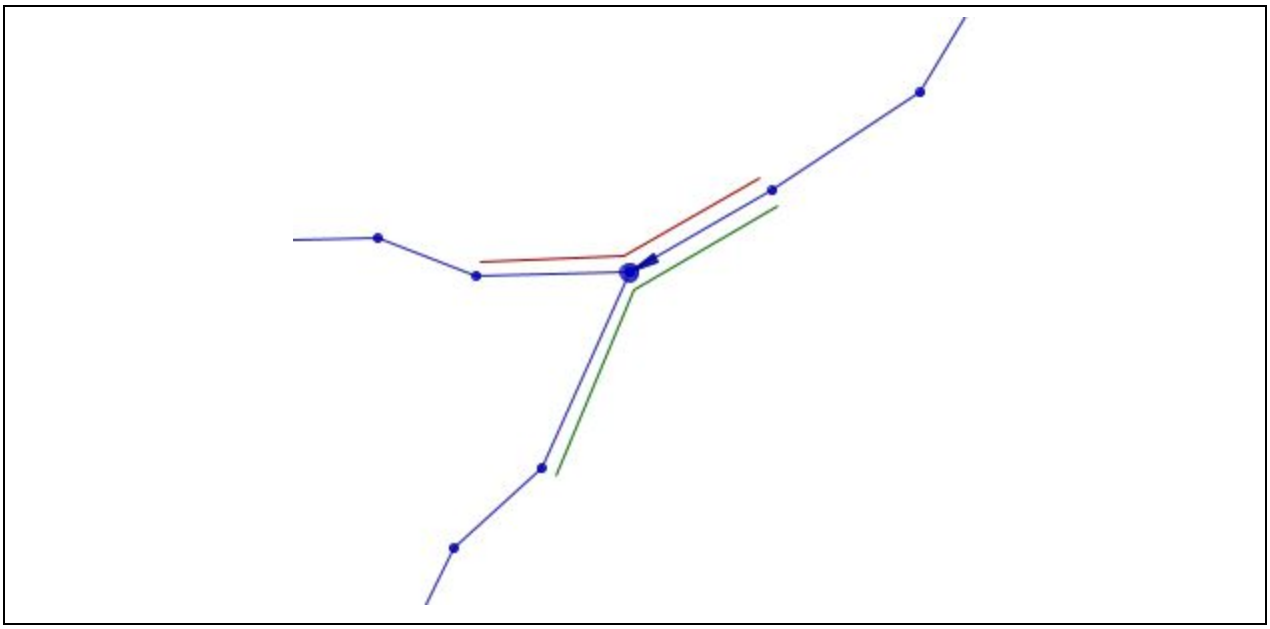
In cases where a channel width can be computed, the channel width along with the angle are used in the determination of the primary flowpath. A rating value between 0 and 1 is generated for both channel width and angle and these are combined using a ratio of 30% angle to 70% channel width to rate each output flowpath with a value between 0 and 1. The largest value is the primary path. The ratio can be modified by changing the value of the rank_channel_weight property.

The angle rating is computed by computing the angle (see details below) and dividing it by 180 degrees. This gives precedence to straighter angles.

The channel rating is computed by computing the widths for all channels, then dividing the individual channel width by this total. This results in wider channels getting higher values.

2.5.2 Computing Angle

Angle is computed using the first coordinates on either side of the flowpath node. In the example below the red and green linestrings represent the angles that are computed.

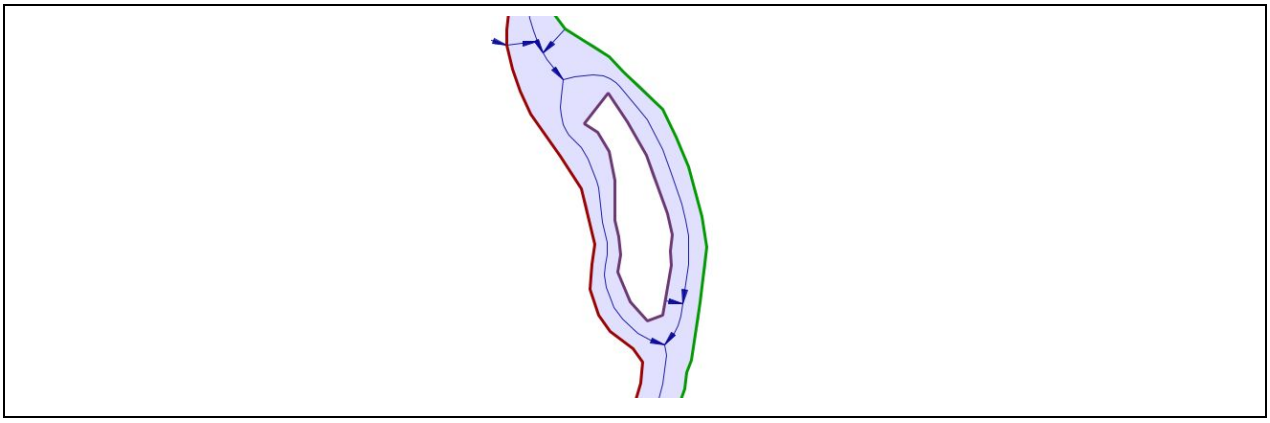


All coordinates are then reprojected to lat/long, then the bearing for the lineSegment represented by the first and second points is computed, and the bearing for the lineSegment represented by the second and third points is computed. The difference between these bearings is the angle for the flowpath.

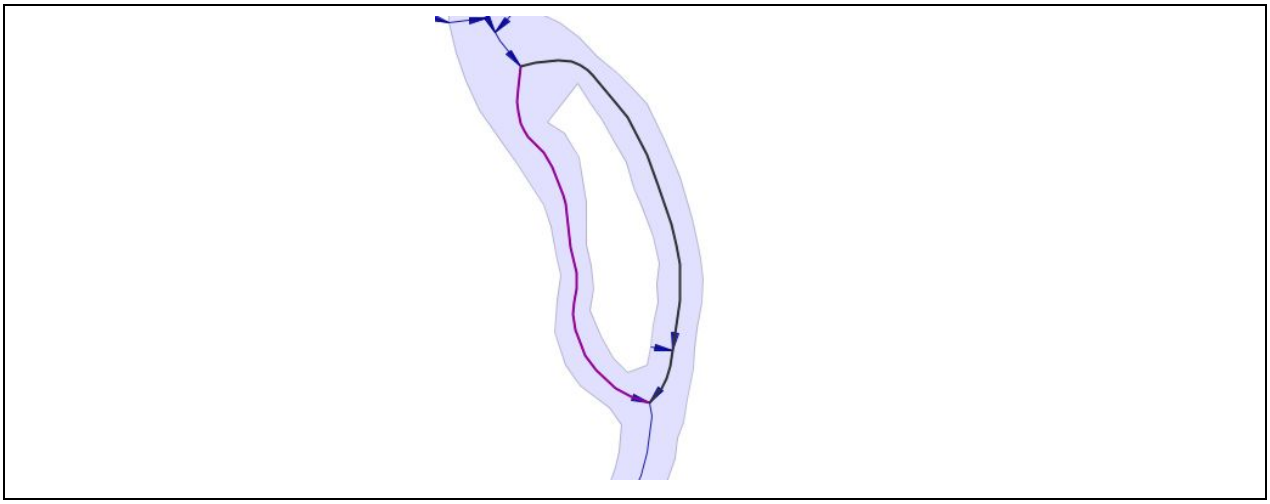
2.5.3 Computing Channel Width

1) The waterbody is broken up into component linestrings. Each interior ring is a component linestring, and the exterior ring is also a component linestring. In cases where the waterbody touches other waterbodies, the exterior ring component is broken into multiple components using these intersection points.

This image shows all the component linestrings of the waterbody coloured in red, green, and purple.



2) For each output, the outflow path is computed. This walks down the stream network through any degree 2 nodes and stops at the first degree three node. Degree three nodes associated with bank flowpaths are treated as degree two nodes. In this example the purple and dark grey lines represent the outflow paths.



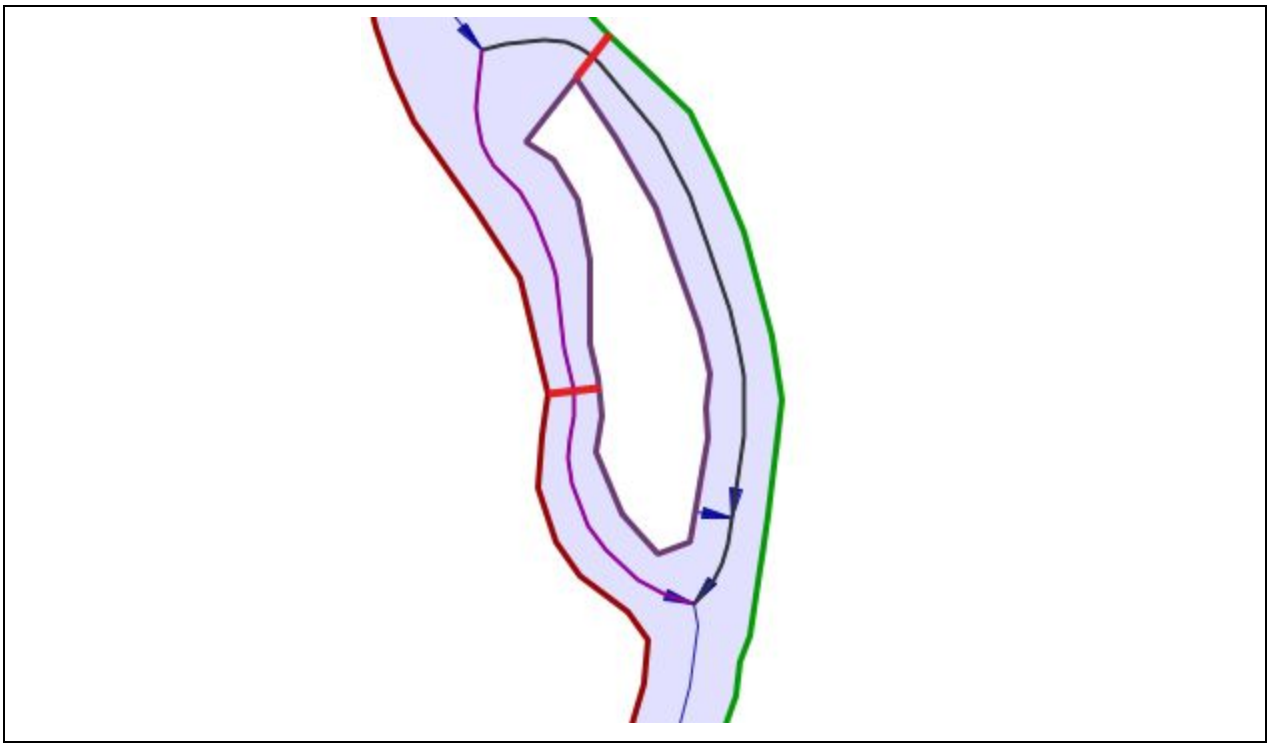
3) The middle point of the outflow path is computed.

4) The two closest waterbody component linestrings to these middle points are found.
In the example above the two closest components to the purple line are the red and purple component, and to the grey line are the purple and green components.

5) For each of the component linestrings found in the previous step the nearest point to the flowpath is found.

6) A line is drawn from the two connecting points and the length of this line is used as the channel width value. This will be calculated in whatever units the raw data is in.

Below, the thick bright red lines represent the linestrings used to compute channel width.



There are cases where the generated lines used for computing channel width are not perpendicular to the flowpaths. Warnings are generated when the angle between the flowpath and generated line segment is < 50 degrees.

3.0 Properties Files

The software uses various properties that can be changed by the user if they desire different results. By default the software includes two properties files (one for planar projections in metres or feet) and one for degree based projections. Each property is described below along with the default values.

Property	Description	Default (m/ft)	Default (degree)
vertex_distance	Used when computing bank input points. The distance away from the center point of a line that the nearest vertex can be before a new coordinate is added along that line. Units in the units of the input dataset.	1	0.001
densify_factor	The factor for densifying the input points (polygon boundaries) to the voronoi process. If this is not high enough the voronoi process will generate	1	0.00001

	lines that cross the waterbody. If this is too high it affects performance and precision. Units are the same as the units of the input dataset.		
simplify_factor	The factor to apply to the Douglas-Peucker algorithm used to simplify the skeleton lines. If set too low skeleton lines will cross waterbody. Units are in the units of the input dataset.	.25	0.000001
minimum_skeleton_length	Length in projection units. The minimum length of skeleton lines. Lines that are smaller than this length will be collapsed. A value of 0 or less will collapse no edges. Don't set this too big or you will end up with incorrect results.	1	0.00001
acute_angle	Angle in degrees. If the angle formed by the lineSegments on either side of an input or output point has an angle less then this value, then these lineSegments are densified at a factor of 10x to prevent issues from voronoi lines going outside the polygon and getting removed and thus no resulting skeleton line connecting this in/out point	30	30
bank_node_distance_offset	Percentage along the line to skip when trying to generate bank skeletons in cases where an existing skeleton already exists. Bank skeletons will not intersect the bank edge for between x% and 1-x%. For example if the value is 0.2 and the bank edge is 10 units long, the bank skeleton line will intersect the bank edge somewhere between 2 and 8 units along it.	0.2	0.2
bank_min_vertex_distance	Distance in projection units. Used when generating bank skeletons in cases where skeletons exist. This will be the minimum distance between vertices for the cases where new vertices are added. If a vertex exists within this distance of where the software wants to create a new vertex it will try to use the existing vertex. There are a few cases where this won't be followed (if the entire line is less then this distance).	0.001	0.000001
dir_short_segement_length	Length in projection units. Used by the directionalizer. Edges that are shorter than this in length are revisited after directionalizing to see if flipping them produces a "better" result. “Better” is based on the average direction of surrounding edges.	15	0.0001
dir_angle_diff	Angle in degrees. Used by the directionalizer.	15	15

	When creating paths if the difference between the start and end angle is greater than this value then the software looks at the start and end angles to determine if flipping the path results in better flow angles (better = straighter outflow, more acute inflow).		
rank_channel_weight	Must be a number between 0 and 1. Weighting of channel width when computing rank. Only applicable for features where channel width can be computed. Flowpath rating= X * channelrating + (1-x) * anglerating	0.7	0.7

Example Properties File:

```
vertex_distance=1
densify_factor=1
simplify_factor=.25
minimum_skeleton_length=1
acute_angle=30
bank_node_distance_offset=0.2
bank_min_vertex_distance=0.001
dir_short_segement_length=15
dir_angle_diff=15
rank_channel_weight=0.7
```

Appendix 2: CHyF Catchment Delineator Details

1.0 Running the Software

To run the software use the .bat (windows) or .sh (linux) files provided as follows:

```
<tool>.bat [OPTIONS] <INFILE> <DEMDIR> <OUTFILE>
```

1.1 Options

The following options can be provided to both commands.

Option	Description
-p	Requires one filename parameter, the name of a custom property file to use.
-c	Requires one integer parameter, the number of cpu cores to use.
-r	Requires no parameter, enables “recovery mode”. The specified output file must already exist and be the result of a previous run which failed or was cancelled. Continues processing blocks that were not previously processed successfully and builds the output catchments. Useful due to the potential for a long running process to fail because of external issues such as network connectivity, etc.

For example:

```
catchment-delineator.bat -p custom.properties -c 4 \
./testdata/Richelieu.32618.fpg.gpkg \
./dem \
./testdata/Richelieu.32618.cd.gpkg
```

1.2 Logging

All logging is written to the console and to a log file named with the date and time: catchmentdelineator-{yyyymmdd}-{hhmmss}.log

2.0 Properties Files

The software uses various properties that can be changed by the user if they desire different results. By default the software includes two properties files (one for planar projections in metres or feet) and one for degree based projections. Each property is described below along with the default values.

Property	Description	Default (m/ft)	Default (degree)
PRECISION	The triangulation code requires a fixed coordinate precision in order to leave room (within double precision math) for calculating divisions of triangles. The precision is fixed at 1 divided by the specified value.	100000	10000000000
BLOCK_SIZE	Data is processed in blocks, each of which is a square of the size specified in the units of the data. Smaller blocks require less memory and process faster, but larger blocks are sometimes required for more consistent results.	10000	0.1
BLOCK_BUFFER_FACTOR	Data outside of each block is included as reference to maintain consistency across block boundaries. The blocks are buffered on all sides by an amount equal to the BLOCK_SIZE multiplied by the BLOCK_BUFFER_FACTOR. Similar to block size, smaller factors require less memory and process faster, but larger factors can be required to maintain consistency, particularly where there are large areas with no flowpaths.	0.5	0.5
CONSTRAINT_HEIGHT	The fixed elevation value assigned to constraint vertices. Should be lower than any DEM vertex, in order to insure water flows to constraints	0.0	0.0
BUBBLE_BIAS_FACTOR	DEM height values are modified to be higher proportionally based on the natural logarithm of the distance from the DEM to the nearest water. Effective Height = DEM Height + BUBBLE_BIAS_FACTOR * log(1 + distance to water)	0.01	0.01
DEM_HULL_INSET	Water Edges are trimmed to be within the DEM hull boundary to prevent effects at the edge of the DEM. DEM_HULL_INSET is the distance the hull is inset before being used to trim the water edges.	2.0	0.00002
SNAP_TOLERANCE	The snap tolerance for enforcing constraints in the TIN surface. If this tolerance is too large, the TIN may be constructed incorrectly (since endpoints of short segments will be snapped to be coincident, thus collapsing triangles which should appear). If the value is too small, there may be robustness problems which appear.	0.005	0.00000005

MIN_DEM_CONSTRAINT_DISTANCE	The minimum distance a DEM point is allowed to be from a constraint segment. DEM points within this distance of water edges are discarded. Enforcing this should prevent Delaunay site snapping from happening during constraint enforcement (which otherwise can produce invalid triangulations).	5.0	0.00005
MAX_SPIKE_BASE_WIDTH	These values are used to identify and remove spikes from the resulting catchment boundaries.	20.0	0.0002
MAX_SPIKE_ANGLE		15.0	15.0
MAX_SMOOTH_ANGLE	These values are used to control the smoothing of the resulting catchment boundaries.	170	170
MIN_LEN_TO_SMOOTH		10.0	0.0001
MAX_SEG_LEN		100.0	0.001

Example Properties File:

```
# WatershedBuilder Properties for meter-based projections
PRECISION = 100000
BLOCK_SIZE = 10000
BLOCK_BUFFER_FACTOR = 1
CONSTRAINT_HEIGHT = 0.0
BUBBLE_BIAS_FACTOR = 0.01
DEM_HULL_INSET = 2.0
SNAP_TOLERANCE = 0.005
MIN_DEM_CONSTRAINT_DISTANCE = 5.0
MAX_SPIKE_BASE_WIDTH = 20.0
MAX_SPIKE_ANGLE = 15.0
MAX_SMOOTH_ANGLE = 170
MIN_LEN_TO_SMOOTH = 10.0
MAX_SEG_LEN = 100.0
```

3.0 Catchment Delineation Process

3.1 Build Hydro Edges

A new feature type is created in the output geopackage called HydroEdges. This feature type has a Linestring geometry, an integer drainageId, and a single character field waterSide. The hydro edge features are created by loading all of the reach flowpaths, water catchments, and Shorelines.

For each reach flowpath, the geometry is taken unchanged, a unique drainageId is assigned, and the waterSide is set to N for neither.

For each water catchment, the outline of the catchment is converted to a set of individual 2-point linestrings. These linestrings are grouped into contiguous sections, separated at any location where another hydro edge touches them, e.g. a flowpath, other water catchment, or shoreline touching the edge of the water catchment. Each of the resulting contiguous sections of linestrings are assigned the same drainageId, and the waterSide is set appropriately to left or right based on which side of the line is the inside of the water catchment.

For each shoreline feature, the individual Linestrings are extracted from any multi-linestring features, and then they are broken up into contiguous sections in the same way as the water catchment outlines.

All of the resulting hydro edge features are written to the output geopackage for later use by the block processing.

3.2 Create Processing Blocks

A second new feature type is added to the output geopackage called ProcessingBlocks. This feature type has an integer identifier, an integer state, and a square polygonal geometry. A square grid based at the origin of the data's coordinate system is overlaid onto the hydro edges and every grid which contains or overlaps a hydro edge is inserted as a processing block feature with a unique identifier and a state of "ready".

3.3 Process Blocks

Each block is then processed individually. Blocks can be processed in parallel if the system has enough memory and processing cores. Note that while the blocks' geometries are disjoint, the block processing area includes a buffer around the block (by default, 50% of the width of the block). Thus the processing overlaps neighboring blocks. Each block is processed as follows.

First, the data is retrieved. This includes all of the hydro edges and all of the DEM points within the buffered block boundary. Next, the polygon hull of the DEM points is calculated. If the DEM is missing in some part of the region, this will ensure that only the area with DEM is processed. The DEM hull is then inset a small amount to prevent any hydro edge features from reaching the edge of the hull, and any hydro edge features which extend beyond the inset hull boundary are trimmed away. If there are any duplicate hydro edges due to data issues, the duplicates are removed.

Next, a Conforming Delaunay Triangulation is constructed from the DEM points and hydro edges, with the hydro edges as constraints. This means that points are added to the triangulation to cause the hydro edges to be edges in the triangulation, while still maintaining the

Delaunay property of the triangulation. The triangulation is then further modified by a process called medial axis refinement. This identifies triangles which have sides touching different hydro edges with different drainageIds, and adds new points and edges to the triangulation to split those triangles. This is to ensure that it is possible to trace a path through the triangulation which is between any neighbouring hydro edges, as would a valid catchment boundary.

Now an adjustment is applied to the height of every DEM point based on the distance from that point to the nearest hydro edge. A KD-tree index is used to identify the nearest hydro edge, and the “bubble bias” is calculated based on the configurable BUBBLE_BIAS_FACTOR and the natural logarithm of the distance. This has the effect of slightly elevating regions that are further from water, causing an artificial ridge at the medial axis between neighbouring hydro edges in very flat areas where identifying the flow direction would otherwise be impossible.

The triangle trickle process comes next. Every triangle is evaluated based on the adjusted height of its corner points to determine to which neighbouring triangle water would flow. Triangles which neighbour a hydro edge are assigned to the drainageId of that hydro edge. All triangles are then connected together along their trickle paths until every triangle is part of some region, with regions identified as either flowing into a particular drainageId, or as “pits”. Pits are local depressions from which the trickling cannot escape. These regions are resolved by identifying their spill point, the lowest point around the boundary of the region, and then merging them into the region which they would spill into.

The outer edges of the final merged regions go through a cleaning process which removes spikes and smooths the lines, which can be very jagged due to being based on the triangulation. The cleaned-up edges of the merged regions are identified as CatchmentConstructionEdges, and are written into a new table in the output geopackage. Due to the nature of the block-by-block processing, edges of regions that are at the bounds of the buffered block are not CatchmentConstructionEdges. The catchment boundary polygons are assembled in the next phase of processing which is done for the entire dataset, allowing for catchments that cross block boundaries.

3.4 Build Catchments

After all of the blocks have been processed, a final process assembles the ECatchment Polygons from the CatchmentConstructionEdges. All of the CatchmentConstructionEdges are loaded from the geopackage and grouped by drainageId. In addition, all water catchments and reach flowpaths are also loaded for reference. The CatchmentConstructionEdges are then assembled into polygons. The Water Catchments and reach flowpaths are used to sort out which sides of the construction lines are the inside and outsides of the resulting catchments, as well as identifying reach catchments and bank catchments. The resulting ECatchment polygons are written to the output geopackage.

Appendix 3: Companion Software

1.0 GeoPackage Reprojector

The GeoPackage Reprojector is a utility provided with the CHyF Processing Tools which can be used to help prepare the input data for processing. It can re-project the vector feature layers in a geopackage file and also reduce the coordinate values to a fixed precision. This tool is not required to be used as the Catchment Delineator already applies a fixed precision to all of the layers it processed, which is required to prevent mathematical robustness issues due to the inability to calculate exact coordinates when using double precision values.

It is recommended that the input data to the flowpath constructor and catchment delineator are in a planar projection using metres or feet (not a geographic projection using degrees). The Catchment Delineator will enforce a fixed precision level no smaller than 1/100000 (this equates to 10 micrometers if meters are used), based on the input property configuration.

This tool can also be used to reproject and/or reduce the precision of other topologically related layers that are not directly part of the flowpath constructor and catchment delineator processes. This will allow the exact topological relationships to be maintained between the other layers and the CHyF layers.

Vector feature layers will be read from the input geopackage file, will have the coordinates reprojected and precision reduced as specified by the command-line options, and will be written to the output geopackage file. Any other data stored in the input geopackage, such as raster layers, will not be transferred to the output geopackage.

1.1 Running the Software

To run the software use the .bat (windows) or .sh (linux) files provided as follows:

```
geopackage-reprojector.bat [OPTIONS] <INFILE> <OUTFILE>
```

1.1.1 Options

The following options can be provided to both commands.

Option	Description
-s	Requires on integer parameter, the EPSG code number of the SRID to which the data should be projected
-p	Requires one integer parameter, the inverse of the precision to which the data should be reduced. Typical values for use with the CHyF tools would be: 100000 - for a planar projection, corresponding to a precision of $1/10^5$

	10000000000 - for geographics (latitude and longitude) corresponding to a precision of $1/10^{10}$

For example:

```
geopackage-reprojector.bat -s 3978 -p 100000 \
./testdata/Richelieu.4617.gpkg \
./testdata/Richelieu.3978.gpkg
```

2.0 gdal_translate

A very useful tool for manipulating DEM data (and other raster data) is “gdal_translate”, available as part of the GDAL/OGR library available from http://gdal.org. Two examples of how it was used to support the CHyF Catchment Delineation process are below.

1. To resolve the issue with predictor-based compression of floating point data (found in data from the USGS), the data must be converted to a GeoTIFF which does not make use of a predictor for floating point compression. The following command achieves this:

```
gdal_translate -co COMPRESS=LZW USGS_1_n49w091.tif USGS_1_n49w091_fixed.tif
```

2. To resample DEM data to a lower resolution to reduce memory usage and processing time (but reduce accuracy to some degree), the following command works:

```
gdal_translate -outsize 50% 50% -co COMPRESS=LZW cdem_dem_073I.tif
../cdem_2/cdem_dem_073I_half.tif
```

See the gdal documentation for more details on the functionality available.

Appendix 4: Screenshots of Inputs and Outputs

The images shown in the following pages are from two of five test areas: The Rainy River Watershed along the Ontario - Minnesota border, and the Kootenay Lake Watershed along the southern border of British Columbia. Rainy is dominated by lakes and wetlands, whereas the Kootenay is quite mountainous. A visual review of the inputs and outputs for these two example areas should give the reader a clear idea of what the software does. In all cases a combination of WMS and WMTS basemaps from Natural Resources Canada is used as background.

The input data consists of basic hydrography, with the lakes and double-line (i.e., polygonal) rivers treated as water catchments and the single-line streams as reach flowpaths. The output adds reach catchments and bank catchments, as well as skeleton flowpaths and bank flowpaths. The additional elementary flowpaths are created by the CHyF Flowpath Generator and the additional elementary catchments are created by the CHyF Catchment Delineator.

As shown here, bank catchments have a yellow tint, indicating that these areas do not contain streams and instead drain directly into lakes and rivers along the banks of these polygonal waterbodies. Primary flowpaths are in blue and secondary flowpaths in orange. Bank flowpaths (all of which are primary flowpaths) are rendered in pastel purple. Markers on the flowpaths indicate the direction of flow. In a couple of cases on Figure 4 it appears that a catchment boundary may touch a stream. This is not the case, as can be verified on a GIS by zooming in sufficiently. In all cases the boundaries meet topological constraints and also comply with the hillslope shape as determined by the elevation data in combination with the position of the hydrologic elements.

The Flowpath Generator provides an option to use existing skeleton flowpaths, in which case only bank flowpaths are generated. However, in the two examples below, all flowpaths were generated.

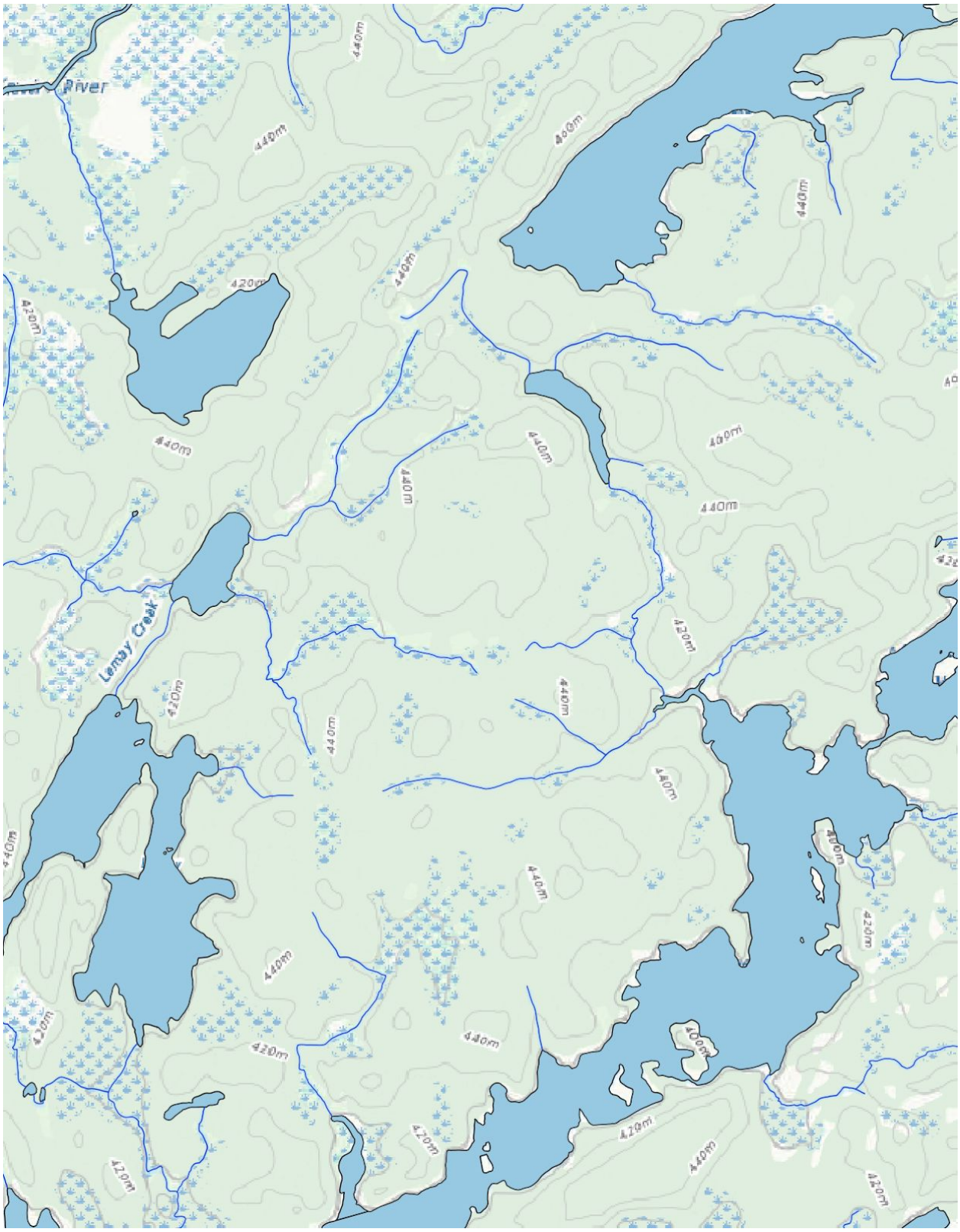


Figure 1: Rainy River area with original hydrography rendered

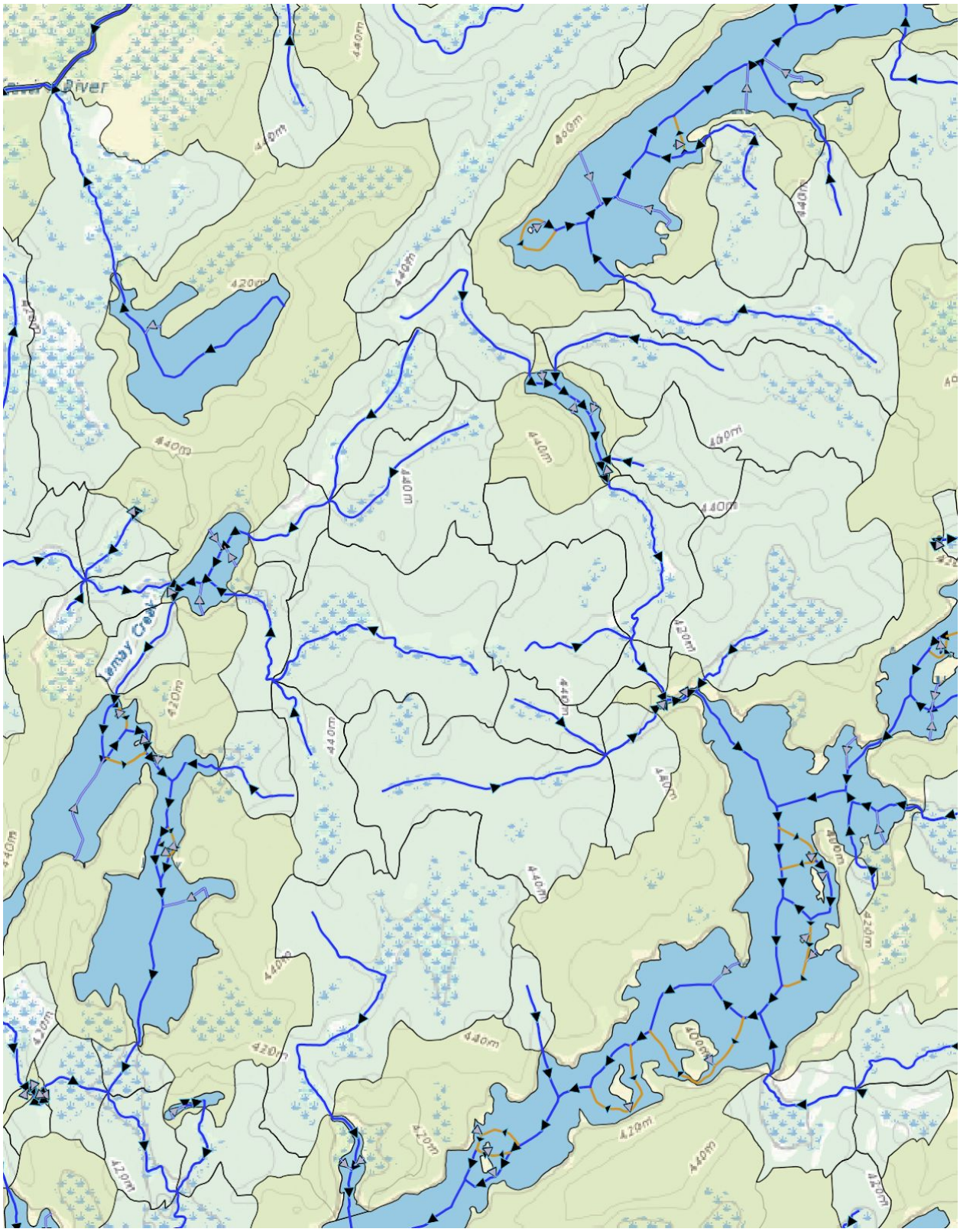


Figure 2: Rainy River area with all flowpaths and catchments displayed

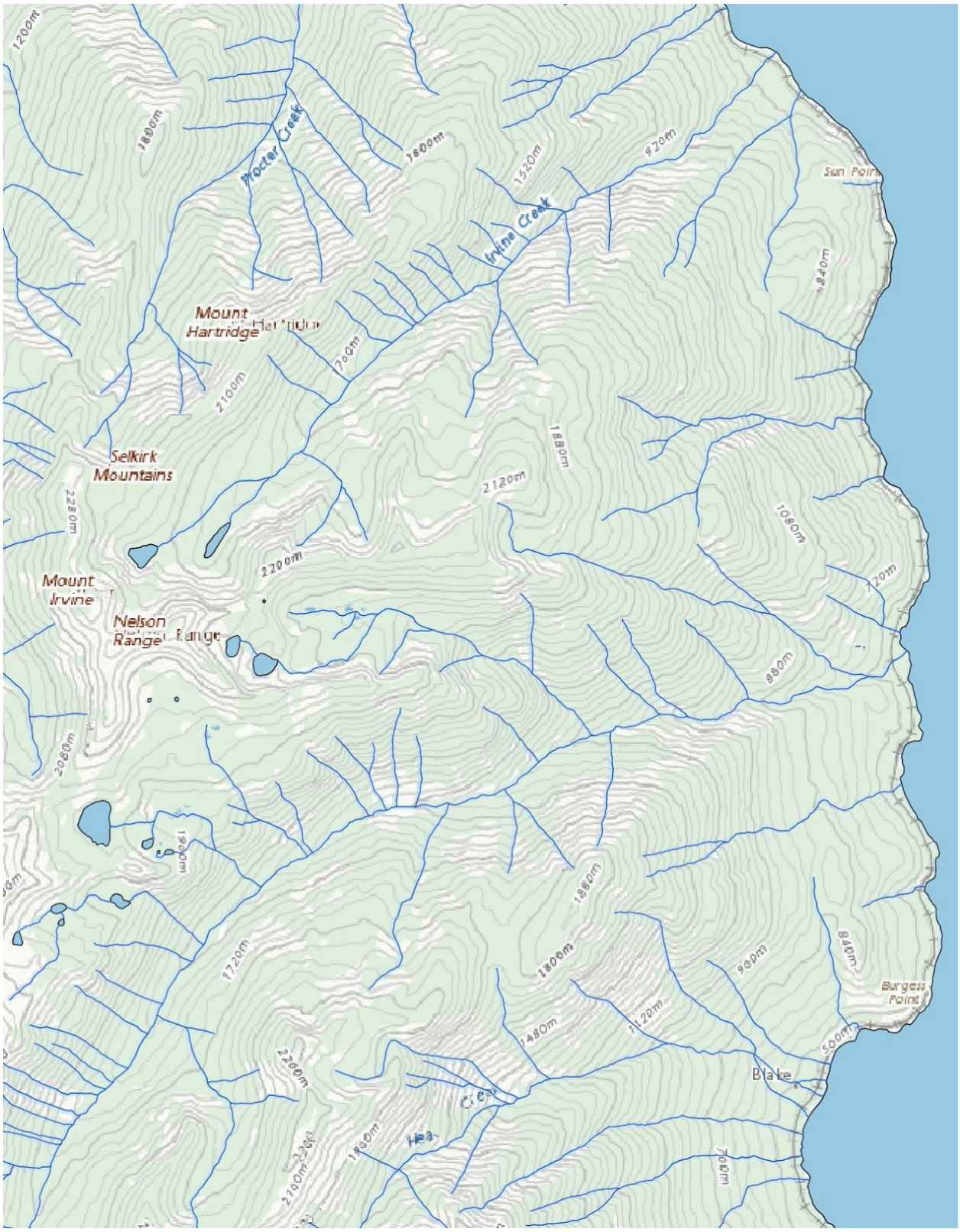


Figure 3: Kootenay Lake area with original hydrography rendered

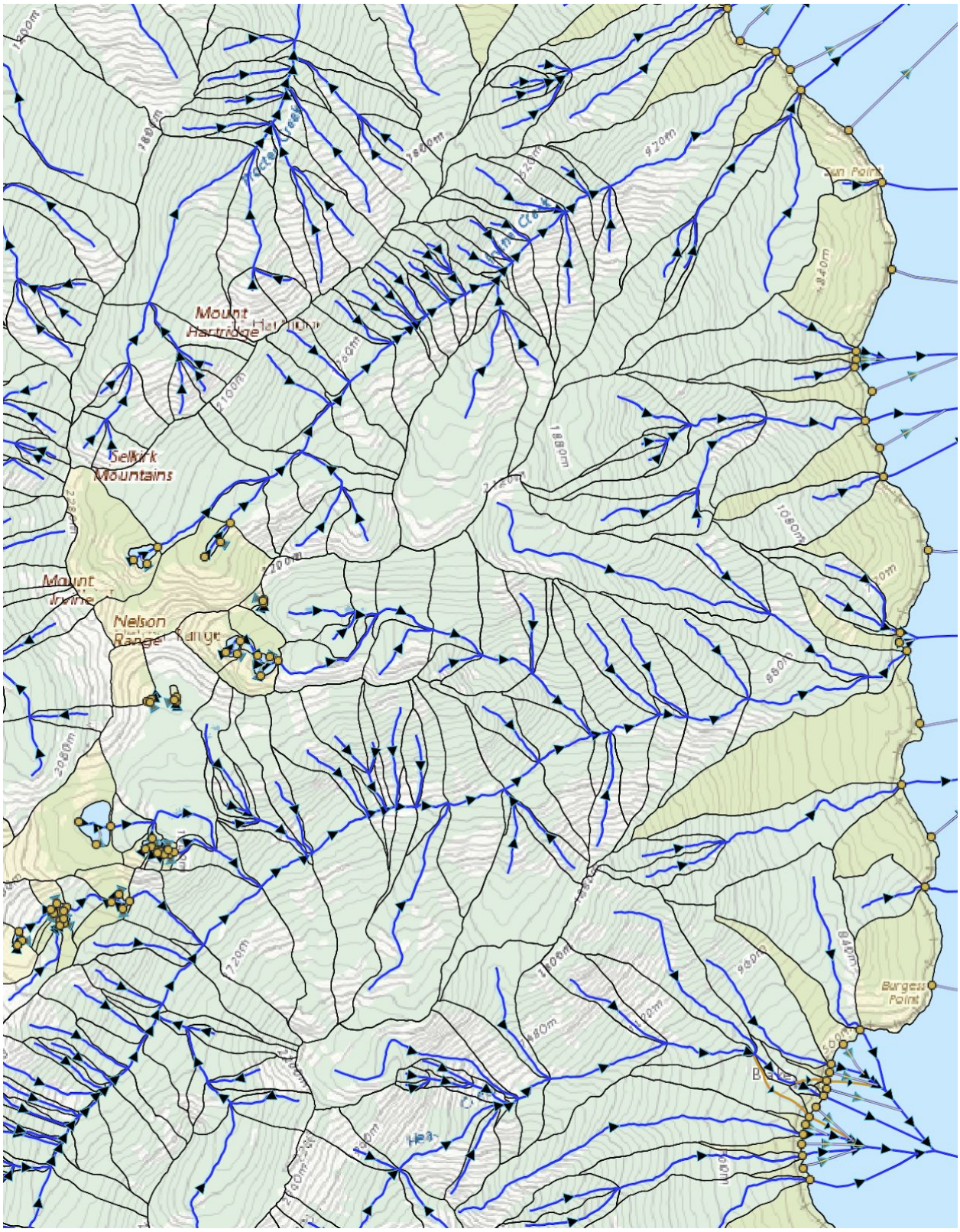


Figure 4: Kootenay Lake area with all flowpaths and catchments displayed