Environnement et Changement climatique Canada
Environment and Climate Change Canada

Technical Note NT-120

LOSLR – IERM habitat models recoding

Sylvain Martin, Guillaume Guénard, Jean-Michel Fiset, and Jean Morin

March 2017

Canada

N° de cat. : xxx

ISBN : xxx

For citation:

Martin, S., G. Guénard, J.-M. Fiset, and J. Morin. (2017). LOSLR – IERM habitat models recoding. Technical Note NT-120, MSC Québec – Hydrology and Ecohydraulic Section, Environment and Climate Change Canada, Québec, prepared for the International Joint Commission study on Lake Ontario and Saint-Lawrence River. 12 pages + Appendix.

Environment and Climate Change Canada

Public Inquiries Centre

7th Floor, Fontaine Building

200 Sacré-Coeur Boulevard

Gatineau QC K1A 0H3

Telephone: 819-997-2800

Toll Free: 1-800-668-6767 (in Canada only)

Email: ec.enviroinfo.ec@canada.ca

Work team

Environment and Climate Change Canada – Meteorological Service of Canada – Hydrology and Ecohydraulic Section

Project management

Sylvain Martin, M.Sc.

Jean Morin, Ph.D.

Programming, data management, and redaction

Sylvain Martin, M.Sc.

Guillaume Guénard, Ph.D.

Jean-Michel Fiset, Jr. Eng.

Table of contents

Figures

Introduction - Problem statement and proposed solution

The environmental performance indicators (PI) for the Lower St. Lawrence River (IERM) are a critical component of the evaluation of water level regulation plans for the Lake Ontario-St. Lawrence River (LOSLR) system and are important to maintain in an adaptive management perspective. These PIs have been mainly created, programmed and used during the Lake Ontario – St. Lawrence River study (2000-2005) and have been used many times since then for new plan assessment and other applications in the river. The 40 PIs are grouped in 14 different components of the ecosystem (eg. floral or faunal resources), each of which being handled by a specific program implementing a model quantifying the effect of discharge or water level modifications on habitat of these floral or faunal resources (habitat models).

The Hydrology and Ecohydraulic Section of Environment and Climate Change Canada (ECCC-HES) was the principal investigator, with several partners, responsible for the creation, validation and the maintenance of the PIs. However, the programing language in which they were implemented (*Visual Basic.Net 2008*) is no longer used by ECCC-HES. Consequently, these PIs are rapidly becoming obsolete and need updating. In the last 6 years, the ECCC-HES has migrated to the *Python* programming language and one of the benefits of this migration is that *Python* is a cross-platform language that can be executed on Microsoft® Windows®, Linux, or any common Operating System (OS), including Canadian Meteorological Center (CMC) operational environment. *Python* is a stable and well documented open-source object-oriented programming language. Furthermore, it is consistent with the operational requirements for the CMC supercomputers and is going to be supported in the near future.

In this context, the ECCC-HES has been tasked to update the Lower St. Lawrence River PIs and this report is detailing the work undertaken on code that has been updated or created including re-designing of object-oriented programs for each existing PI, recoding the 14 programs (habitat models) from *Visual Basic.Net* to *Python 2.7*, including quality verification and execution coordination.

Moreover, the integration process for new discharge/water level time series (new regulation plans), including time series pre-analysis check, coordination of the different calculation steps, verification spatial results outputs, and global results management are steps that are presently time consuming and has been significantly optimized to reduce time and efforts required to achieve these tasks. Consequently, this report also describes the creation of additional programs aiming at enhancing the integration of new time series into the IERM internal format, comparing new time series (among existing groups) with existing ones, creating spatially explicit results for verification, and storing global end results in a format consistent with the adaptive management needs.

As a consequence of the IERM models recoding project and the *Python* projects repository constraints, this report will also detail the new file and folder repository structure and its benefits for model portability to different OS.

The final product will be a “nearly automated system” encompassing: 1) the integration of new time series, 2) PIs calculation, and 3) production of final results. We expect that it will significantly speed-up execution time and minimize efforts required to achieve the assessment of new regulation plans.

Project milestones

PI re-designing and recoding

Object-oriented classes re-design and PIs recoding to Python

In the former implementation of habitat model programs, functions that were common to all models calculation were grouped in module (code file) that model programs can use at runtime. No common function was created for data insertion and all data necessary to run models were retrieved from a database. This last step is time-consuming at runtime since complete datasets have to be transferred from the database to the computer where programs are executed.

Here, a new object-oriented code design has been applied to re-implement habitat models for the Lower St. Lawrence River PIs in *Python 2.7*. To achieve this, we first created a parent class that contains functions common to all 14 models. This parent class includes functions retrieving information from a configuration file in order to specify the correct time series, IERM Grid, physical environmental data, triggering data, etc. and also functions inserting global and spatially explicit data in a database. Figure 1~~Figure 1~~ shows a schematic view of the ModeleHabitat class (parent class) containing common functions and the ModeleSpecific class (child class), which represents any of the 14 specific habitat models, inheriting attributes and functions from the mother class ModeleHabitat while implementing specific functions of a given habitat model. At runtime, the parent and the child classes are consolidated as one object containing all attributes and functions of both classes. Hence, any ModeleSpecific class is also a ModeleHabitat class. This behavior is called polymorphism in object-oriented programming.

Each model having its own data needs, the consolidated class retrieves all data listed in a model-specific configuration file, namely the IERM grid, time series, wave input data and frequencies and hydrodynamic input data. Individual classes have been created for each of these datasets, which integrate data locally stored in binary files instead of retrieving them from a database, thereby saving time involved in data access and transfer through network. At runtime, these new classes or objects become part of habitat models. The habitat model is thus composed with these datasets rather than inheriting from them.

Regrouping common code in a parent class improves efficiency since only a single copy of the common functions needs to be developed and maintained.

Execution time is also improved since steps involved in requesting and retrieving data from a database were replaced by binary data access from files located on the computer where programs are executed.

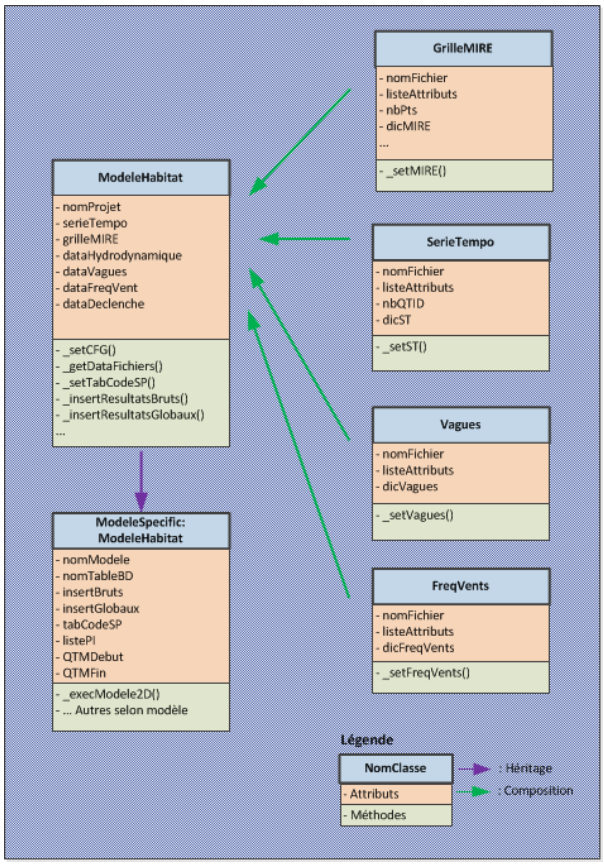

Figure 1 : Object-oriented schematics describing the new habitat models implementation. The child class ModeleSpecic inherits (purple arrow) attributes and functions from the parent class ModeleHabitat. The resulting consolidation contains datasets like GrilleMIRE, SerieTempo, Vagues, and FreqVents (green arrows).

In order to check whether the recoded models rendered consistent results with respect to the former Visual Basic.Net implementation, they were tested against time series HDD, S1DD, S2DD, HBV710, S1BV710, S2BV710, and HBV7985. Each combination of PI model and time series, has been executed on 41 years and results, mainly expressed as habitat surface area (hectare). They were summed by IERM grid division and year and were compared to results from the former implementation. Maximum differences lower than 0.005% were observed between implementations. Maximum differences were observed for two models: that for the least bittern and the yellow perch (0.004%). On the other hand, very small differences were observed in the wind wave data calculation process and they had very little impact on cattails probabilities of occurrence, especially when they are close to the threshold values for a presence. Although differences are very small, they may nevertheless entail noticeable changes in the habitat surfaces predicted by models like that for the least bittern and the yellow perch models since they use results from the cattails model as an input. As these models are showing relatively low surface area values, a small difference in predicted habitat surface may cause large consequences, especially when differences are expressed as percentage.

Nevertheless, differences remain small and we regard the Lower St. Lawrence River IERM habitat models implemented in *Python* as providing equivalent results as their counterparts implemented in *Visual Basic.Net*.

Global results management

In the *Visual Basic.Net* version of the system, global results (ie. summation of potential habitat predicted for a given combination of time series and year) were retrieved after execution using *SQL* requests applied to database tables warehousing results for each IERM grid points and model. This step was time-consuming since all 14 tables had to be visited and some manual data management was still necessary to gather and format information. In the *Python* version, code has been added to habitat models and global results are calculated while PI programs are being executed and potential habitats surface areas are summed for each IERM grid point at runtime. Once a model has processed all points, global results (summation) are stored in a global results database table containing substantially less data (raw data are also stored into the database for validation purposes). Finally, a single *SQL* request sent to the database allows us to retrieve all global results for a given time series in order for them to be compared to the ones already processed.

This additional coding substantially reduced the time and efforts involved in gathering and managing results.

Coordination of execution

The execution has been analyzed for 14 models considering potential dependencies among PIs and a new coordination strategy has been applied. Following that strategy, all PIs that depend on results from either the wetlands or the cattails model were grouped separately for execution. This two-strain re-arrangement is meant to be executed using two processes running simultaneously, the first beginning with the wetlands model and the second with the cattails model.

In the former version, models were executed one after another using a single process. Using two parallel processes instead of a single one achieved an important improvement here in reducing computation time.

Recoding programs supporting models

Improving new time series integration

Water levels and discharge time series insertion and update

The *Visual Basic.NET* program formerly integrating time series provided by the IJC into the IERM computation environment has been re-implemented in *Python 2.7*. This program retrieves time series data from a text file, inserts this information into

a database, and updates water levels calculated at given stations along the Lower St. Lawrence River for each time step.

The former *Visual Basic.Net* version of this application was doing the same work and no improvement in execution time was noted. On the other hand, the provided input text format may involve a short data management step since format differences may occur between time series provided.

Inter-annual quarter-monthly average discharge graphs

Inter-annual quarter-monthly (QM) average discharge graphs are used as a preliminary checking step prior to running models. This step allows users to compare the new time series with others that had already been processed, namely the HDD time series (historical inputs using the 1958DD regulation plan), which has been used as a basis for comparison from the beginning of the LOSLR study.

Before the present re-implementation, inter-annual quarter monthly averages had to be retrieved from the database and transferred to a spreadsheet (Microsoft® Excel®) where graphs were manually created.

In the present version, a *Python 2.7* program generates the time series comparison graphs automatically. Averages are retrieved from a database and graphs are created using the *matplotlib Python* package. Figure 2~~Figure 2~~ shows a typical output graph of inter-annual QM average discharges at Sorel for the HDD, HBV7, and HBV7a time series.

This step brought a notable improvement to the previous workflow since the manual data management and graph creation steps have been automated.

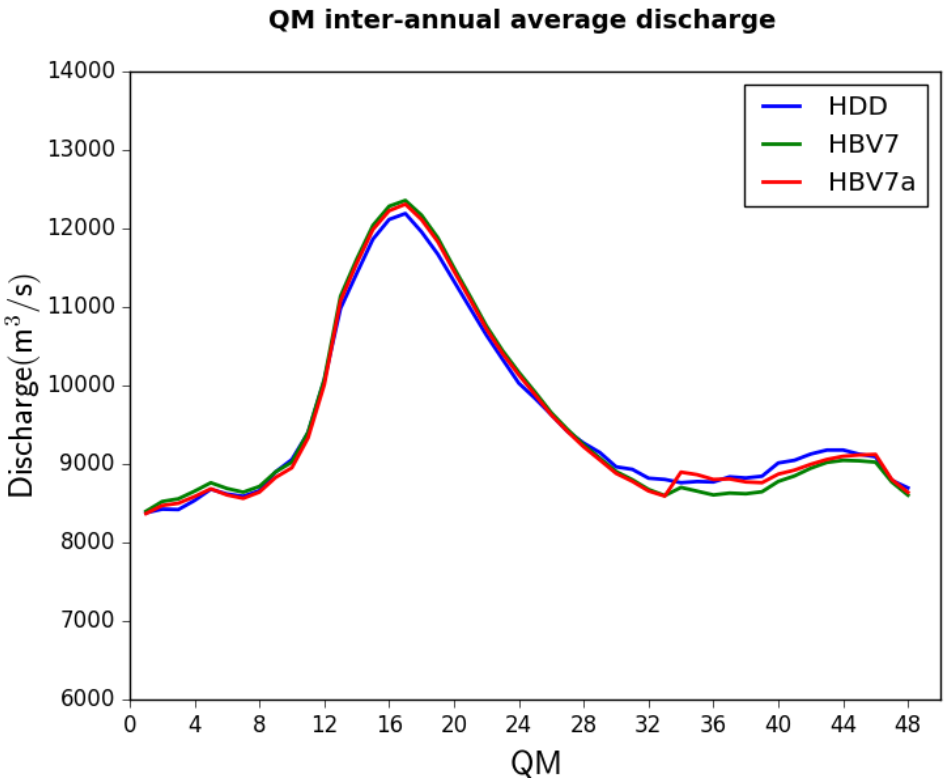

Figure 2 : Inter-annual quarter-monthly average discharge at Sorel for three time series (HDD: blue, HBV7: green, and HBV7a: red).

Time series statistical preliminary check (pre-check) analysis

Another program has been developed using *R* language embedded into *Python* to statistically verify potential anomalies in newly integrated time series. This program will be used as a pre-check step before running models in order to avoid time spent in executing an erroneous time series. A distance metric was developed to assess to what extent time series differ from one another. A Principal Coordinates Analysis (PCoA) was calculated using that distance metric and used to visualize the resemblance or difference between the existing water discharge time series. Time series pertain on one five classes (H: Historic, S1: Stochastic 1, S2: Stochastic 2, S4: Stochastic 4, and CC: Climatic Change). Classification software was also implemented to compare new time series with known ones and assign them to one of the five aforementioned classes to the new time series. Time series that significantly differ from any of the know classes of water discharge time series are classified as being of an unknown type. To achieve this, the program is provided with a minimum probability threshold below which a time series is considered as not being part of a group. New time series are considered as being of an unknown type when not found to pertain to any known group. A PCoA ordination diagram showing the location of the new time series with respect to existing ones is given as an output. Users therefore have the possibility to judge, on the basis of the discharge graph, ordination diagram, and classification results,

whether the new time series she or he submitted is worthwhile for further computation. As an example, Figure 3~~Figure 3~~ shows the position of a new time series position among all existing ones. The new time series called “TEST” is a copy of the “HBV7” (Historic supplies using the BV7 management plan) to which 100 m^3/s has been added to Lasalle discharge for each QM. This modified series is still grouped among the Historic water supply class but with an out of group probability of 0.4671 (instead of 0).

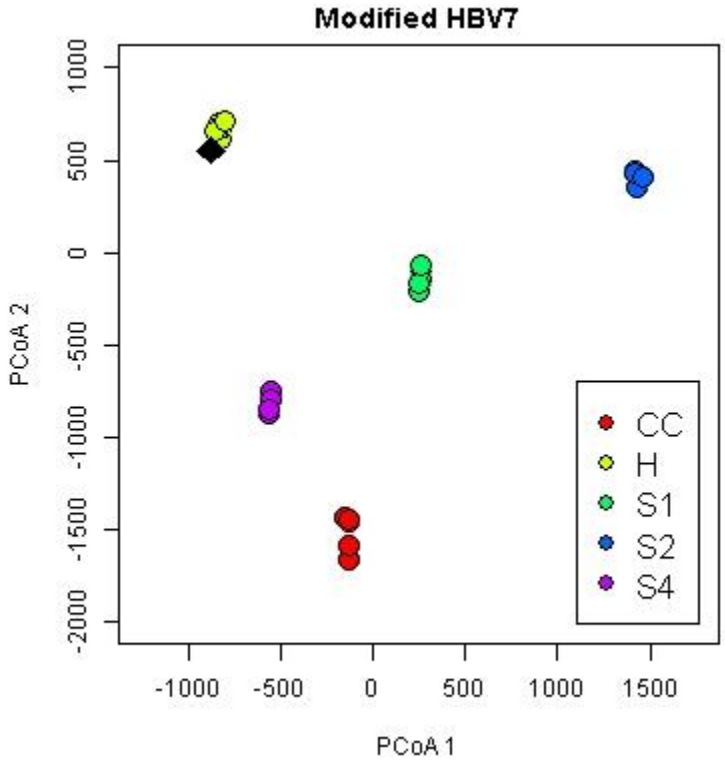

Figure 3 : PCoA graph showing the position of a new “TEST” time series (black dot) among existing ones.

Since there was no such statistical pre-check counterpart in the previous version, no processing time improvement can be observed here but we expect that time wasted processing erroneous time series will be largely reduced.

It is noteworthy that this program depends on the presence of the *R* language and environment in the execution environment and on the *Python* module *rpy2*, which has to be installed in order to execute R functions from *Python*. Both the *R* language and environment and the *rpy2 Python* module are cross-platform, open source, well-maintained, and widely available software.

Improving results output

Spatial data output

Spatially-explicit model data output is important since maps can be used to validate the correct execution of models or judge their behavior by observing spatial shifts in time according to water level variations. Maps can be created for a given combination of PI, time series, and year for which results are stored in the database.

In order to build such maps in the *Visual Basic.NET* environment, it was necessary to download data from the database through a GIS software for each combination of PI, time series, and year. Depending on the GIS software used, this step can be time-consuming since all specifications of PI, time series, year, file location, and name might have to be manually input. In the new version, a *Python* program has been created to automatically retrieve spatially-explicit results and create GIS files from them. Users can also specify a suite of combinations and the new program will create as many GIS files as requested. Specification regarding PI, time series, and year are user-provided and results are automatically transferred in a GIS file located in a PI-specific folder. This program requires the installation of the *GDAL/OGR* executables in the programming environment. This software is freely-available, cross-platform, and open-source. Output format is '*ESRI* shapefile'.

A significant improvement is also observed here considering the time saved not having to create the GIS files manually.

Global results output

As seen earlier in this document, global results are now stored in a database table instead of being retrieved from 14 raw data tables. In order to provide the IJC with data comparing time series results, a *Python 2.7* program was created to retrieve global results from the database and transfer them to an Excel® spreadsheet having the same format than the one in current usage to compare time series results. Users simply provide a list of time series she or he wishes to compare. This program requires the *pandas* package installation in the *Python* environment.

An important improvement is also observed here since this step eliminates time used to retrieve data from the database and manually format the *Excel*® spreadsheet.

Code isolation and structuration

Python has its own code modules (file containing code) importation system for modules included in a defined *Python* project repository. In order to build a clear repository structure, i.e., one compartmenting models and supporting programs and respecting IERM project rationales, the IERM folder repository structure underwent significant changes in order to adapt to Python importing system while new classes and functions was being developed.

In order to respect the project logic, it has been decided to create a single main folder called **IERM2D** with five specific sub-folders: the **bin** folder containing project's Python code; the **cfg** folder containing configuration files (one for each model); the **log** folder containing execution logging files; the **npz** folder containing the essential binary data files, and the **results** folder containing spatially-explicit model results, inter-annual QM average discharge graphs, and global results files.

The **bin** folder contains a sub-folder for habitat models (**ModeleHabitat**) and another one for supporting programs. The **ModeleHabitat** sub-folder contains the parent class also named **ModeleHabitat** (described earlier) and 14 other sub-folders containing model-specific code. The **CodeUtilProjet** sub-folder, located in the **bin** folder is the repository for supporting programs. See *Appendix 1* for more details regarding the repository file and folder structure and contents.

As a consequence, code has been transferred, structured and isolated from the IERM development environment. It is now free from dependencies and self-contained in its own file and folder system. This situation allows models to be easily transferred across platform, regardless of the OS (*Microsoft® Windows®, or Linux*) as long as *Python* installation includes all necessary packages (*matplotlib, panda, r2py, etc*.) and can access the necessary executables (*R, OGR/GDAL*).

Conclusion

All Lower St. Lawrence River habitat models (or environmental PIs) included in the IERM environment hosted by the ECCC-HES have been migrated to the *Python 2.7* programming language. Models' object-oriented design has been improved and encapsulates common functions and attributes into a parent class that each specific model inherits.

Otherwise, many programs involved in the assessment of regulation plan through habitats have also been recoded or created *de novo* into *Python 2.7*, namely programs aiming at integrating water level time series in the IERM system, build inter-annual QM average discharge graphs, compare a given time series to existing ones, retrieve spatially-explicit model data from a database and transfer them to GIS files, and select global results from a database and transfer them to an *Excel*® spreadsheet.

Re-design, re-coding, creation of additional supporting programs and execution coordination re-arrangement contributed to substantially improve the assessment of new regulation plans by decreasing execution time and avoid spending time manually handling data in the many steps involved from time series reception to global results output. While the assessment of a new regulation plan analyzed over 41-year worth of data took about 48 hours on a computer equipped with a 3.2 GHz processor and 16 GB RAM with the former *Visual Basic.NET*-implemented IERM system, it now requires about six hours; an eight-fold improvement in computation time.

Finally, the repository structure has also been re-designed to respect the IERM project rationales and habitat model structure and to comply with the *Python* importing system. Given that *Python* code is portable across OS such as *Windows* or *Linux*, and that the IERM project is now self-contained in its own folder, it can now be easily transferred and executed in a *Linux* environment like CMC's operational environment or environments that IJC partners might have access to as long as a connection to a database is possible. The improvements described here brought the IERM system to a nearly-automated, efficient, cross-platform, and OS-independent state.

Next steps

The next step of the recoding project would be to give access to the IERM2D to a larger audience. This would be possible by the IERM2D integration into a shared environment or to develop a web-base access to the *Python* code and the associated database. Both of these potential solutions have the benefit to give access to the habitat models and supporting programs execution to other study stakeholders thus promoting modelling through the IERM2D. Otherwise, these solutions have two main drawbacks. First, as the implementation is presently inside ECCC's firewall, a new development implementation would have to be created in an environment that all study stakeholders may have access to. Also, given the size of the database in use at ECCC, the task to re-implement it is achievable but would require a substantial amount of work. This database is supported by a *DataBase Management System* (*DBMS*, *Oracle*®) and stores, only

for this project, nearly 2 terabytes of data mainly composed of the Digital Elevation Model (DEM), the IERM2D grid, physical variables (hydrodynamic and wind wave), time series, and habitat models results.

Appendix

Appendix 1 – Habitat models data comparison

Percentages of difference between habitat models processed with the former VisualBasic.Net and the new Python 2.7 implementations. Habitat models are summed over a 41-year period and the IERM Lower St. Lawrence River division.

			Percentage difference						
Ecosystem component	PI name	IERM Code	HDD	S1DD	S2DD	HBV710	S1BV710	S2BV710	HBV7985
Wetlands	Water	EAU	<0,00%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetlands	Forest	Forêt	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetlands	Forested swamp	MARBO	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetlands	Shrubby swamp	MARBU	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetlands	Deep marsh	MP	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetlands	Shallow marsh	MPP	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetlands	Deep marsh influenced by wave	MP_V	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetlands	Prairie meadow	PH	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Cattail	Narrow leaf cattail	Typha_A	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Cattail	Broad-leaved cattail	Typha_L	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (reproduction)	Northern pike	ESLU	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (reproduction)	Yellow perch	PEFL	<0,000%	<0,002%	<0,004%	<0,004%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Lake sturgeon	ACFU	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Northern pike	ESLU	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Brown bullhead	ICNE	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Pumpkinseed	LEGI	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Largemouth bass	MISA	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Golden shiner	NOCR	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Spottail shiner	NOHU	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Yellow perch	PEFL	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Sauger	STCA	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Fish (feeding)	Walleye	STVI	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Muskrat	Muskrat	ONZI	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Narrowleaf water-plantain	ALGR	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Common coontail	CEDE	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Canadian pondweed	ELCA	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Water Star Grass	HEDU	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Eurasian watermilfoil	MYSP	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Sago pondweed	POPE	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Richardson's pondweed	PORI	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Valisneria	VAAM	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Submerged plants	Submerged plants density	Density	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	American bittern	BOLE	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Veery	CAFU	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Black tern	CHNI	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Marsh wren	CIPA	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Common moorhen	GACH	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Swamp sparrow	MEGE	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Song sparrow	MEME	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Sora	POCA	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Pied-billed grebe	POPO	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Wetland birds	Virginia rail	RALI	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Dabbling ducks	Migration	ANPL_MIG	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Dabbling ducks	Elevage	ANPL_ELEV	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Endangered species	Least bittern	IXEX	<0,004%	<0,001	<0,001	<0,004%	<0,000%	<0,000%	<0,004%
Endangered species	Briddel shiner	NOBI	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Endangered species	Sand darter	AMPE	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%
Endangered species	Map turtle	GRGE	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%	<0,000%

Appendix 2 – Repository structure description

Alphabetically-ordered list of all files and folders into which the IERM2D project is defined and self-contained. For convenience, all .pyc (Python compiled) and __init__.py (defining Python project folders) files are not shown.

Folder or file	Content
IERM2D	Main folder
│	
├──bin	All Python code
│ │	
│ ├──CodeUtilProjet	Supporting programs code
│ │ │	
│ │ ├──AnalyseST	Time series analysis and graphs
│ │ │ FlowTSPreCheck-Aux.R	R code file containing PCoA functions
│ │ │ FlowTSPreCheck.py	Compares a new time series
│ │ │ FlowTSPreCheck.rda	Contains R functions for PCoA
│ │ │ graphST.py	Time series graph tool
│ │ │ QSorelTS.cls	Contains existing time series classes
│ │ │ QSorelTS.cnd	Contains PCoA configuration data
│ │ │ QSorelTS.dat	Contains existing time series
│ │ │ RPCoATS.py	Embedded R code into a Python object
│ │ │ SerieTempoHQ.py	Time series insertion and update
│ │ │ TSTBC.dat	Contains the new time series
│ │ │	
│ │ ├──GlobalResults	
│ │ │ makeGlobalResultsFile.py	Global results output file program
│ │ │	
│ │ ├──MetadonneesModeles	
│ │ │ MetadonneesLOSLR.py	Models metadata
│ │ │	
│ │ └──Spatial	
│ │ ResultsFromBD.py	Spatially explicit data output program
│ │ tabPt2shp.py	Text to GIS transformation tool
│ │	
│ └──ModeleHabitat	Habitat model main folder
│ │ mystruct.py	Data structuring tool used by all
│ │ startModele.py	Starts sequencially all models
│ │ startModeleMH.py	Starts the Wetland dependant models
│ │ startModeleTypha.py	Starts the Cattail dependant models
│ │	
│ ├──DB	
│ │ DBUtil.py	Database connection manager
│ │	
│ ├──Declenche	
│ │ declenche_v2.py	Triggering data manager
│ │	
│ │	
│ │	
│ ├──FHS	
│ │ FHS_v2.py	General binary file manager
│ │	
│ ├──Grid	
│ │ grid_v2.py	IERM2D grid binary file manager
│ │	
│ ├──MHPoolAmen	

MHPoolAmen_v2.py	Managed pool depth manager
Modeles	Folder grouping specific models
ModeleHabitat.py	Mother class ModeleHabitat
AMPE	
AMPE.py	Sand darter model
Canards	
Canards.py	Waterfowl model
DensiteSub	
DensiteSub.py	Submerged plants density model
ESLU	
ESLU.py	Northern Pike model
GRGE	
GRGE.py	Map Turtle model
IXEX	
IXEX.py	Least bittern model
MH	
MH.py	Wetland model
NOBI	
NOBI.py	Bridle shiner model
OiseauMH	
OiseauMH.py	Wetland birds model
ONZI	
ONZI.py	Muskrat model
PEFL	
PEFL.py	Yellow perch model
PlanteSub	
PlSub.py	Submerged plants model
PoissonEte	
PoissonEte.py	Fish feeding ground model
Typha	
Typha.py	Cattail model
Profondeur	
coteMob_v2.py	Natural pools depth manager
poolAmen.py	Managed pools depth manager
profQT.py	QM water level calculator
SerieTempo	
serieTempo_v2.py	Time series binary file manager
Substrat	
substrat.py	Substrat data manager

│ │ talus.py	Bank data manager
│ │	
│ ├──utils	
│ │ MQt.py	QM transformation functions
│ │ MUtil.py	Basic functions used by all models
│ │	
│ ├──UtilSol	
│ │ utilSol.py	Land use data manager
│ │	
│ └──VarPhys	
│ varPhys_v2.py	Physical variables binary file manager
│	
├──cfg	All configuration files (1 per model)
│ AMPE_2D.cfg	
│ CanardBarboteur_2D.cfg	
│ DENS_SUB_2D.cfg	
│ ESLU_2D.cfg	
│ GRGE_2D.cfg	
│ IXEX_2D.cfg	
│ MH_2D.cfg	
│ NOBI_2D.cfg	
│ OiseauMH_2D.cfg	
│ ONZI_2D.cfg	
│ PEFL_2D.cfg	
│ PlSub_2D.cfg	
│ POISSON_2D.cfg	
│ Typha_2D.cfg	
│	
├──log	Logging files (1 per model execution)
│ MH_HDD_1951_2000.log	Logging example for wetland with HDD
│	
├──npz	Binary files
│ CoteMob_130513.npz	Natural pools data file
│ CoteNat_280711.npz	Managed pools data file
│ Grid_CMI3_20160414.npz	IERM2D grid data file
│ MHPoolAmen_CMI3_20160414.npz	Managed pools wetland types data file
│ ST_CMI3_HBV710_20160501.npz	HBV710 series data file
│ ST_CMI3_HBV7985_20160501.npz	HBV7985 series data file
│ ST_CMI3_HBV7_20160501.npz	HBV7 series data file
│ ST_CMI3_HDD_20160501.npz	HDD series data file
│ ST_CMI3_S1BV710_20160501.npz	S1BV710 series data file
│ ST_CMI3_S1DD_20160501.npz	S1DD series data file
│ ST_CMI3_S2BV710_20160501.npz	S2BV710 series data file
│ ST_CMI3_S2DD_20160501.npz	S2DD series data file
│ VarPhys_280513.npz	Physical variables data file
│	
└──results	Results folder
│ GlobalResults_20161006.xlsx	Global results output file example
│	
├──AMPE	Spatial Sand darter data
│ AMPE_HDD_1972.dbf	GIS files example showing Sand darter habitat distribution that would occur in 1972 using the HDD regulation plan
│ AMPE_HDD_1972.shp	
│ AMPE_HDD_1972.shx	
│	
├──Canards	Spatial Waterfowl data
│	
├──Densite	Spatial Submerged plant density data

├──	
├── ESLU	Spatial Northern pike data
├──	
├── GRGE	Spatial Map turtle data
├──	
├── IXEX	Spatial Least bittern data
├──	
├── MH	Spatial Wetland data
├──	
├── NOBI	Spatial Bridle shiner data
├──	
├── OiseauMH	Spatial Wetland birds data
├──	
├── ONZI	Spatial Muskrat data
├──	
├── PEFL	Spatial Yellow perch data
├──	
├── PlSub	Spatial Submerged plant data
├──	
├── PoissonsEte	Spatial Fish feeding ground data
├──	
├── ST	Time series graphs
│ St_20161116.png	Example of inter-annual QM graph
├──	
└── Typha	Spatial Cattail data

Pour des renseignements supplémentaires :

Environnement et Changement climatique Canada
Centre de renseignements à la population
7e étage, édifice Fontaine
200, boulevard Sacré-Cœur
Gatineau (Québec) K1A 0H3
Téléphone : 1-800-668-6767 (au Canada seulement) ou 819-997-2800
Courriel : ec.enviroinfo.ec@canada.ca