ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

eawag
aquatic research

MASTER THESIS

A prelude to emulation for *flood prediction*

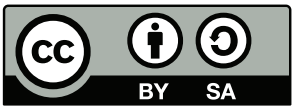
Author:
Sebastiano RUSCA

Supervisor:
Dr. Juan Pablo CARBAJAL
Dr. Jörg RIECKERMANN

Examiner:
Prof. Dr. Peter Molnar

ETH Zürich - Dept. of Civil, Environmental and Geomatic Engineering
Chair of Hydrology and Water Resources Management

March 29, 2018



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

A PRELUDE TO EMULATION FOR FLOOD PREDICTION

Authored by (in block letters):

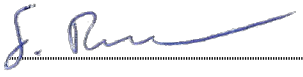
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
RUSCA	SEBASTIANO

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 23.03.2018	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Abstract

MSc. ETH Environmental Engineering

A prelude to emulation for
flood prediction

In recent decades, with the growth of computational power and the development of sophisticated and accurate hydrodynamical numerical models, simulations have become a powerful tool to make hydrological predictions. Emulation, the approximation of detailed numerical simulators with simplified *ad-hoc* models, reduces the excessive computational burden of the numerical simulators,thus allowing the development of near-real time applications. After an introduction to the basic principles of emulation, this thesis proposes an innovative early flood warning system. This alert system is constructed by emulating a numerical simulator solving the Shallow Water Equation for overland flow. Application of the illustrated methodology to real-world situations will permit to emanate timely alert for the implementation of measures targeted at reducing the damages caused by floods.

Acknowledgements

I first of all would like to thank my two supervisors, Dr. Juan Pablo Carbajal and Dr. Jörg Rieckermann, and the examiner of my thesis Prof. Dr. Peter Molnar. Their support was fundamental during the course of my whole master thesis. In particular, I want to thank Juan Pablo for all the time spent with me, staying at the eawag until late at night just a couple of days before Christmas, in order to completely debug the package fswof2d. Or again, spending hours at the telephone with me to discuss last details of the emulator a couple of days before my intermediate presentation. It was just great to pursue my thesis under your supervision; for this just THANK YOU. Thank you too Jörg Rieckermann, your punctual comments to my draft allowed me to find the guideline I couldn't find so far. The help of Vasilis at the very beginning of my thesis, when still struggling to understand the functioning of my simulator was very encouraging; I am grateful for this. I also would like to thank Joao, providing me the topography for the simulation that we sadly could not run because of lack of time, and with it many useful suggestions. Eawag is a great place, because it is full of great people. I am very glad that I had the opportunity of spending my thesis period there. Thank you to my office mates, to the running group, and to all of the people who made eawag such a special place.

This thesis was also possible thanks to all of the support I received from the outside. I want to thank my parents for always being present when I needed. And my sister and her farm, for offering me the right diversion when I wanted to knock off. I am particularly grateful to my good friend Gionata, offering me the moral and technical support I needed when feeling lost and confused.

Last but not least, a special thanks goes to my "WG". Donato and Gloria; Gloria and Donato, who accompanied from near all of my thesis, with its ups and downs; and were always there to make me laugh when I needed.

Finally, a special thought goes to the developers of all of the open softwares that allowed the realization of my thesis. You are doing great! A huge thanks GNU Octave, FullSWOF_2D and Inkscape.

Contents

Chapter 1

Introduction

Floods are one of the major socio-economic risks for the population. According to the European Environmental Agency (EEA), floods, together with wind storms, are the natural hazards which cause the highest economic loss in Europe (European Environment Agency, 2013). In the next decades, climate change is expected to increase flood risk at global scale (Milly et al., 2002; Hirabayashi, Kanae, et al., 2008; Hirabayashi, Mahendran, et al., 2013). In Europe, the magnitude and direction of trends is projected to vary between northern and southern regions (Dankers and Feyen, 2009; Alfieri et al., 2015; Thober et al., 2018). Still, several studies reported important expected changes in the structure of European precipitation, with particular intensification of short-duration extreme rainfall (O. Christensen and J. Christensen, 2004; Zolina et al., 2010; Westra, Alexander, and Zwiers, 2013) which are likely to increase the danger of extreme flash floods. At the same time, increasing warming temperature are modifying snow dynamics in mountainous regions, altering the nature and processes of floods (Köplin et al., 2014; Hall et al., 2014; Berghuijs, Woods, and Hrachowitz, 2014). Recent studies illustrated ongoing changes in hydrological regimes at continental-scale (Blöschl et al., 2017; Stahl et al., 2012). These hydro-climatic shifts might dampen our capability to predict correctly the occurrence and risk of future flooding on infrastructures designed on past hydrological regimes (Milly et al., 2002). Detection of these vulnerable infrastructures and implementation of timely alert could potentially reduce the damage and fatalities associated to extreme flood events.

In recent decades, with the growth of computational power and the development of sophisticated and accurate numerical models, simulations have become a powerful tool to make hydrological predictions. The degree of accuracy of these numerical models depends often on the complexity of the hydrological processes representation and the spatio-temporal resolution of the simulations. Unfortunately, for very complex and accurate simulations there is a price to pay: the computational cost is very elevated; a single simulation can take from minutes up to days and even weeks. As a consequence, the creation of simulation-based early flood alert warning system is hindered by the computational burden of simulations.

Emulation of these numerical models provides a tool to reduce the model evaluation to a few seconds, which allows generation of near real-time predictions. Emulators, also referred as "surrogate models", are ad hoc data-driven models that mimic the behaviour of a detailed numerical simulator. When properly constructed, they can reduce the computational cost of simulations by several orders of magnitude at the expense of a slight decrease in accuracy (Carbajal, Leitão, et al., 2016).

In this thesis, two case studies are presented which show the potential of emulation of a numerical simulator solving the Shallow Water Equation (SWE). Chapter 2

provides an introduction to the concept of emulation (2.1), a description of some regression techniques (2.2) and presents the numerical simulator solving the SWE (2.3) used in the case studies described under Chapter 3. The first case study (3.1) illustrates the construction of a *mechanistic emulator* to estimate the water depth above a weir as a function of discharge in the channel. The potential of emulation in substituting the realization classical laboratory experiments with numerical ones can be inferred. Case study 2 (3.2) presents a possible workflow to develop an *emulator-based* early flood warning system. Due to the generalization and abstraction of the proposed methodology, its testing in real specific case studies should be easily implementable.

The whole thesis was pursued using *open softwares* and is licensed under the CC BY-SA 4.0 license. The simulations from which the datasets used to build the emulators were extracted, were run with FullSWOF_2D-v.1.07.00. All data manipulation and plotting was performed with GNU Octave. To ease the interaction with the simulator the GNU Octave package *fswof2d* was developed. This is available online and distributed under license GPLv3+. The URLs where this, as well as the rest of the thesis' material can be downloaded, can be found in Sec. A.1 of the Appendix.

Chapter 2

Methods

2.1 Emulation

In the recent decades, the increasing computational capabilities of computer processors has enlarged the applications of numerical simulators and opened the possibility to an increase of the simulation resolution and accuracy. In many engineering fields, the level of detail of these simulators has grown to the point of substituting the realization of laboratory experiments. Despite the numerous advantages provided by these numerical simulators, the elevated computing time of the simulations remains a major drawback, especially for development of real-time applications or testing/optimization of different simulation conditions/designs.

A possible solution to this problem is provided by emulators; approximation models which try to reproduce the behaviour of the detailed simulator but with minor computational costs. Construction of appropriate emulators allows huge speedups at the expense of unnecessary simulation informations and accuracy (Carbajal, Leitão, et al., 2016).

Emulators are task-specific, built to answer a specific question. If the research question changes, a new, ad-hoc emulator must be built. Let us make an example from the hydrodynamics field, where the research target is to know the water depth time-series at a specific point in a river channel knowing the (input) hydrograph at the inlet of the channel. Several numerical simulations could be performed based on different input hydrographs, saving the respective simulated water depth time series at the location of interest. Then, an emulator could be built to learn the relationship between the hydrograph and the observed water-depth time series. This would allow an estimate of the water depth time-series from a specific input hydrograph to be obtained without running a new simulation. If a new task would request the prediction of the flow velocity time-series at the same location, the previously constructed emulator would not be able to provide an answer, because trained only with the pair water depth-input hydrograph. If this new task has to be accomplished many times, it could make sense to build a new emulator in order to reduce the overall computational time. The above example shows that emulators can be space-specific (e.g. water depth time-series at a specific point) or time specific (e.g. water depth profile along a channel at specific instant).

When running hydrodynamical simulations, the simulator goes through a series of *internal states* that are often not needed for a specific application. The structure of such simulators is said "fan-out/fan-in". (Carbajal and Rieckermann, 2017). Fig. 2.1 schematizes this peculiarity: in hydrodynamics SWE must be solved over the whole simulation grid, although the output of interest might just be the water depth simulated at a chosen grid cell.

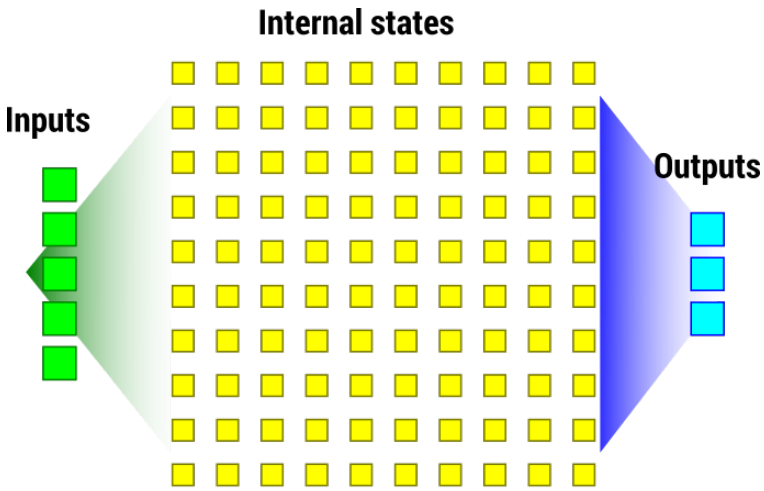


FIGURE 2.1: Fan-out/fan-in structure of typical simulators used in hydrodynamics, solving the SWE over a grid (Carbajal and Rieckermann, 2017).

In contrast to the numerical simulator, an emulator bridges the internal states: it goes from the inputs to the desired output, without going through all of the internal states. Fig. 2.2 depicts this relationship bridging the internal states.

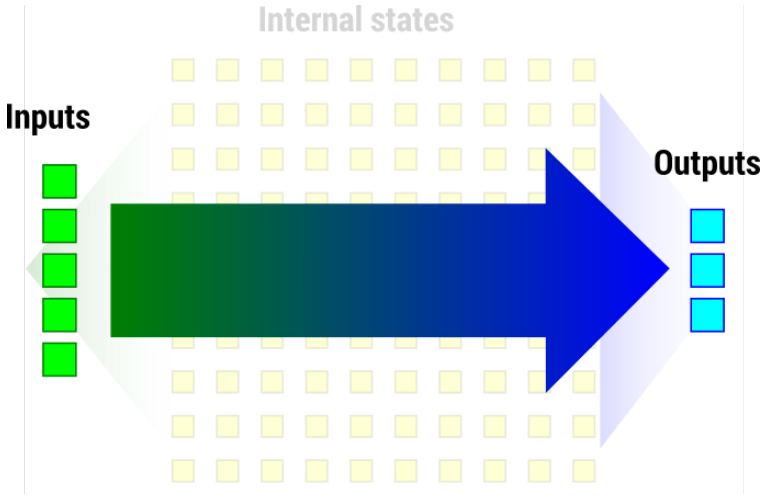


FIGURE 2.2: Emulator functioning: a direct relationship linking the inputs to the outputs, independent from the internal states (Carbajal and Rieckermann, 2017).

The direct functional relationship between the simulated inputs and output sets is established using some regression techniques (see Sec. 2.2).

Emulators can mainly be divided into two groups: *mechanistic* emulators and purely *data-driven* emulators. Mechanistic emulators are constructed by encoding prior knowledge of the system and by constraining the regression problem. The first case study (see Sec. 3.1) illustrates this type of emulators, by using a partial theoretically equation (the *weir equation*) to estimate the water depth over a weir with specified design.

On the other hand, data-driven emulators try to uncover the relationships within the inputs-output sets by exploiting the potential of machine learning (ML) algorithms. In this case, no knowledge about the process generating the inputs-output

sets is incorporated in the emulator. The second case study (see Sec. 3.2) applies this type of emulation.

2.2 Regression and interpolation

A key step when building an emulator is establishing the functional relationship between the simulated inputs-output sets. This statistical exercise correspond to solve an interpolation or regression problem. If the simulated inputs-output sets are considered exact, an interpolation is preferred: this means that the functional relationship must pass through all data points. In this thesis, simulations performed within case studies 1 and 2 are considered exacts: every simulation run with the same inputs produces the same output. As a consequence, the emulators presented in case studies 1 and 2 aim at interpolating the simulated inputs-output sets. In case study 1, this could not be done with the *weir equation*, given the inflexibility of this model.

A large variety of methods exist to solve interpolation or regression problems. The choice may depend on the number of inputs (dimensions of the inputs space), on the amount of observations available to establish the relationship or on previous knowledge about the considered system. Within this thesis three different types of interpolation techniques, implemented within GNU Octave, were used.

- 1D-linear interpolation (`interp1 (X, Y, XI, "linear")`)
- 1D-cubic spline interpolation (`interp1 (X, Y, XI, "spline")`)
- Gaussian Processes (GP) interpolation (package gpml)

Of the three techniques, GP interpolation is the most flexible; it was therefore applied to build the emulator of case study 2.

A GP is fully specified by its *mean* ($m(\boldsymbol{x})$) and *covariance* function ($k(\boldsymbol{x}, \boldsymbol{x}')$) (Rasmussen and Williams, 2006) and is commonly written as:

$$f(\boldsymbol{x}) \backsim \mathcal{GP}\left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{2.1}$$

Both the mean ($m(\boldsymbol{x})$) and covariance functions ($k(\boldsymbol{x}, \boldsymbol{x}')$) may depend on a set of hyperparameters. These hyperparameters define properties of the interpolation surface between the data points, like its smoothness. By choosing specific mean and covariance functions, or setting constraints on the hyperparameters, GPs allow to encode prior knowledge in the interpolation. This way the number of observations needed for an accurate intrapolation (predicted behaviour between two interpolated observations) can be reduced. Extrapolation, namely predicting the behaviour of the process beyond the available observations, should also be more accurate when prior knowledge is embedded.

The package gpml offers a set of mean, covariance and likelihood functions as well as various prior distributions and inference methods readily available to the user. Ad-hoc mean and covariance functions can be generated by the user as composition of provided base functions (e.g. sum, product or power of functions).

Tuning of mean and covariance functions' hyperparameters to obtain the best fit can be done using the function *minimize*. This minimizes the negative logarithm of the *marginal likelihood* $p(\boldsymbol{y}|\boldsymbol{X})$ (the probability of observing the output $\boldsymbol{y}$ given the inputs $\boldsymbol{X}$).

2.3 FullSWOF_2D

The numerical simulator which was emulated in the two case studies presented in Chapter 3 is FullSWOF_2D-v.1.07.00. FullSWOF (Full Shallow Water equations for Overland Flow) is an *open source* software solving the SWE over a regular uniform grid using finite volume method (FVM) (The FullSWOF Team, 2018).

Water can enter the simulation domain through the boundaries (*imposed discharge*) or with the rain. Inflow discharge can be varied independently for every boundary, but remains fixed in time. On the contrary, rainfall is applied uniformly to the simulation domain, but can be varied in time. If the option rain is activated, a *rainfall file* specifying the rain hyetograph has to be provided.

The outflow of water occurs through the boundaries by setting a *Neumann* boundary condition. Infiltration is implemented by means of a modified version of the Green-Ampt equation, correcting for values of the infiltration capacity tending to infinite. The infiltrated amount of water is stored in the soil and does not leave the computational domain.

In order to run simulations, FullSWOF_2D needs at least the following inputs:

1. **Topography**: text file specifying the topography of the gridded simulation domain. This is done specifying the coordinates (x and y) of each grid cell and the respective elevation (z) with an x-y-z format.
2. **Parameters**: text file setting the values of the simulation parameters. These include:
   - Number of cells (x and y directions) (Nx, Ny)
   - Simulation duration (t_{max})
   - Number of intermediate states saved (N_{states})
   - Domain length (Lx)
   - Domain width (Ly)
   - Boundary conditions specifications
   - Various settings for the numerical schemes
   - Different physical parameters (friction coefficient, initial soil saturation, soil thickness, soil hydraulic conductivity, maximal infiltration)
3. **Initial conditions**: text file defining the initial conditions of the simulation. It specifies the *water depth* and *flow velocity* for every cell of the domain.

Every simulation produces the following outputs:

1. **Initial state**: simulations' results at every node at the beginning of the simulation ($t = 0\,\mathrm{s}$). This corresponds to the initial conditions file.
2. **Final state**: simulations' results at every node at the final state of the simulation ($t = t_{max}$).
3. **Intermediate states**: simulations' results at every node for every one of the N_{states} saved. It begins with the initial conditions and ends with the final state.
4. **Water budget**: how much water was lost through the boundaries, how much infiltrated, and how much is present on top of the topography.

5. **Parameters copy**: copy of the simulation's parameters used to produce the given output.

In order to generate the required input files, interaction functions were developed for GNU Octave. These were grouped into a package which is distributed[1] under the GPLv3+ license.

[1]The *fswof2d* package can be downloaded at: `https://bitbucket.org/binello7/fswof2d/`

Chapter 3

Case studies

Case study 1

3.1 A mechanistic emulator: fitting the *weir equation*

As a first case study to apply the acquired knowledge, it was decided to build an utterly mechanistic emulator. The main goal of this case study is to try to fit the *weir equation* to simulated data instead of to experimental data. Here we essentially rediscover the way science has always been done: by observing, measuring and trying to discover mathematical relationships. The most significant difference is the fact that the experimental set-up is completely computer-built.

The *weir equation* is a partially theoretical equation that provides an estimate of the discharge Q over a weir as a function of the water depth above the weir itself (h_w). The equation can be derived from the Bernoulli equation under certain assumptions (Bos, 1989) and can be found in many different forms. The form used here is the one proposed by *Francis* (Walcott, 1907):

$$Q = C \cdot L \cdot h_w^a \,, \quad \text{usu. } a = 3/2 \tag{3.1}$$

Where the empirical coefficient C corrects for the assumption of absence of viscous effects and uniform velocity distribution, L is the length of the weir (perpendicular to the channel) and h_w is the water height above the weir.

In addition to the parametric model (*weir equation*), two non-parametric local techniques, namely *linear interpolation* and *cubic spline interpolation*, were used to intrapolate between the simulated data points.

3.1.1 Brief experiment description

For this experiment, simulations were run in a flat rectangular channel. A weir with a trapezoidal cross section is located at the channel half-length. As initial condition the upstream side of the weir was filled with water up to the weir crest. At the domain top boundary a constant inflow discharge was set, while at the bottom boundary water could freely outflow. As the simulation runs, the inflow water flows down the channel, overflows the weir and leaves the domain through the lower boundary (see Fig. A.1). After some time the simulation reaches the *steady-state* conditions: inflow, discharge over the weir and outflow have the same magnitude and the water height above the weir has stabilized. At this point the value h_w was extracted and was paired with the discharge value Q generating it. 25 experiments were conducted

with Q linearly spaced in the range $[0.1, 10]\,\mathrm{m^3\,s^{-1}}$. All simulated pairs (Q, h_w) constitute an *inputs-output* sets to which the weir equation was fitted.

In order to ensure the convergence of the simulator solution, and therefore the quality of the experimental results, a *grid convergence study* was performed prior to the experiment. For this, the simulation with the highest discharge was repeated with successive grid refinements. The value of the variable of interest, water height, was then compared between the different simulations to find at which grid resolution the solution stabilizes.

3.1.2 Material and methods

Generating the topography

The topography used for running the simulations was generated in GNU Octave and represents a flat channel of 40 m length with a weir placed at its midpoint, at 20 m distance from the top boundary. The channel cross section is a rectangle of 4 m width and the weir has a trapezoidal shape. Fig. 3.1 shows the geometry of the channel. The weir has a crest width of 2 m and therefore belongs to the *broad-crested* class. The C coefficient for broad-crested weirs with vertical walls and 2 m crest width varies between 1.36 and 1.53, depending on the water height h_w (Brown et al., 2009). We therefore expect a value close to this for our experiment.

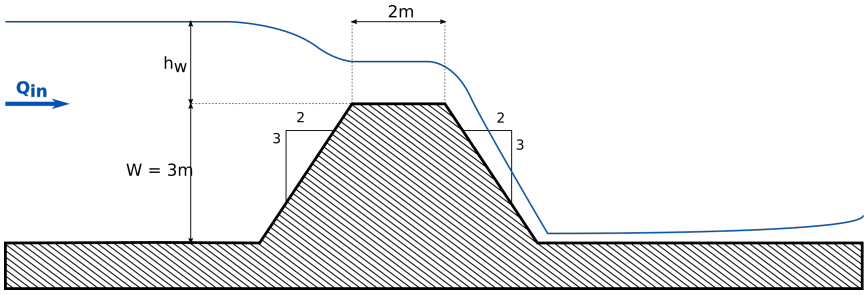


FIGURE 3.1: Geometry of the weir used for the experiments.

Setting up the simulations

After generating the topography to be used for the experiment, the simulation parameters were defined.

The simulation was done with no rain and no infiltration. This way, water just enters the system through the top boundary and leaves it through the bottom one. For this a *Neumann* boundary condition was chosen at the bottom, while an *imposed discharge* boundary condition was set at the inflow boundary. This boundary condition needs two values to be specified: the magnitude of the inflow and an imposed water height, which is used under supercritical flow conditions. The 25 inflow discharges used were obtained with the following GNU Octave code:

```
nQ  = 25; # number of experiments
Qin = linspace (-0.1, -10, nQ); # Qin values [m3/s]
```

where, by definition within FullSWOF_2D, the minus sign indicates that water steadily *enters* the domain. The imposed height was set to 3 m, corresponding to the weir height.

The channel topography presents no banks. Wall boundary conditions were set for the lateral boundaries in order to produce the rectangular cross section. The wall boundaries extend infinitely high, preventing the water from overflowing. Simulations were run for $t_{max} = 200$ s and 200 intermediate states were saved, giving a time resolution of the simulation output of 1 s. For the channel a uniform Manning roughness coefficient of $0.03\,\mathrm{s\,m^{-1/3}}$ was used.

Since infiltration was not active for the simulations, all of the parameters relative to the soil model did not have to be set. The number of cells in x, and in y direction were defined according to the grid convergence study explained here below.

Conducting the grid convergence study

The chosen grid resolution influences accuracy of simulation results as well as simulation runtime. In order to find the appropriate grid resolution, a grid convergence study was performed. Successive refinements were tested in order to find out at which resolution the solution converges. A criterion has to be established, in order to decide when the solution has satisfactorily converged. If the Grid Convergence Index (GCI) is used, then a GCI lower than 5 % for the last refinement is usually taken as a criterion (Ali, Doolan, and Wheatley, 2009).

For the mesh study 7 simulations were run by varying the grid resolution (Nx, Ny) only. Topography and parameters used are those mentioned in the two preceding sections. Squared cells were used and the inflow discharge was set to the highest discharge value of the experiment, namely $10\,\mathrm{m^3\,s^{-1}}$. This discharge is the one generating the highest flow velocities. If convergence under these conditions is reached, then convergence for lower discharges is also assured. Tab. 3.1 summarizes the simulations' main characteristics.

TABLE 3.1: Summary of simulations runtime, grid resolution and other related parameters for the mesh convergence study.

#	Nx	Ny	Lx [m]	Ly [m]	dx [m]	dy [m]	Q [$\mathrm{m^3\,s^{-1}}$]	runtime [s]
1	2	20	4	40	2.00	2.00	10	0.00
2	4	40	4	40	1.00	1.00	10	0.07
3	8	80	4	40	0.50	0.50	10	0.60
4	20	200	4	40	0.20	0.20	10	9.00
5	40	400	4	40	0.10	0.10	10	69.60
6	80	800	4	40	0.05	0.05	10	537.72
7	100	1000	4	40	0.04	0.04	10	1048.40

Fig. 3.2 shows the free surface profiles of the 7 simulations at steady-state conditions along the channel axis. It can be noticed that for $Ny = 20$, 40 and 80 the water depth is visibly higher than for further refinements, and its shape quite varying. This means that the solution has not converged yet. Results for $Ny = 400$, 800 and 1000 are very close, since the lines are almost overlapping. $Ny = 200$ shows a similar profile shape, but the height is still diverging significantly from the successive refinements. To better observe the variations between the different refinements, the h value convergence at the weir crest was studied.

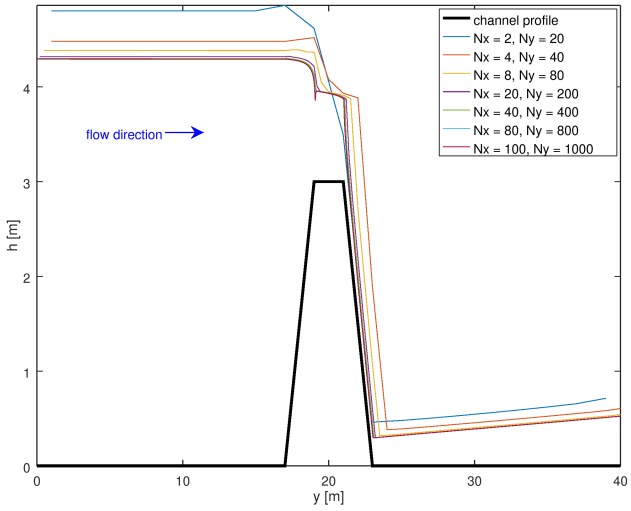


FIGURE 3.2: Longitudinal free surface profiles at steady-state conditions for the 7 different grid resolutions. Convergence of the solution for finer grids can be observed.

In Fig. 3.3 the percent variation in the value of h can be observed. Measured absolute h values at the weir center can be observed in Fig. A.2 in the Appendix. At the 4^{th} refinement the percent variation is smaller than 0.01 %. This variation is small enough to assert that the solution has satisfactory converged. For the experiment it was therefore decided to use $(Nx, Ny) = (40, 400)$ corresponding to a grid resolution of 0.10 m in both x and y directions. The simulation runtime of $\approx$ 1 h10 min, required at this resolution is still quite acceptable.

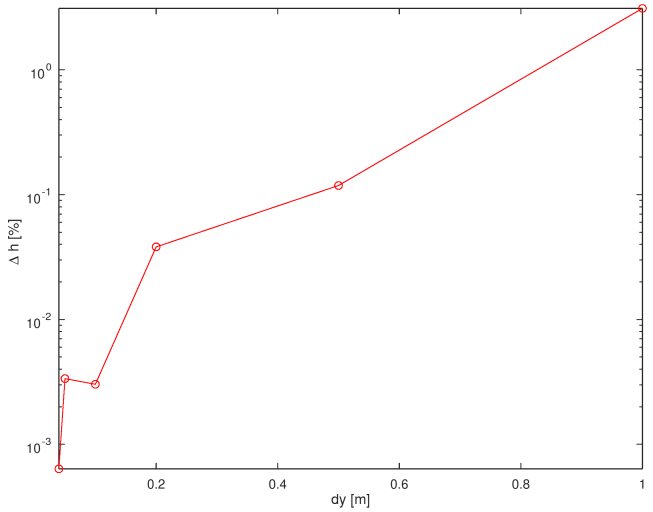


FIGURE 3.3: Percent variation of the measured h at the weir midpoint between the given dy and the previous one.

Extracting the dataset

The weir equation computes the discharge over the weir as a function of the water height h_w under steady-state conditions. These are reached after $\approx 50\,\text{s}$ of simulation. After this time, small oscillations of the water surface are still present. To remove these, a time averaged free surface from $t = 100\,\text{s}$ to $t = 200\,\text{s}$ was computed. Fig. 3.4 presents the free surface profiles of the 25 experiments at steady-state conditions. The three lowest profiles correspond to the simulation runs with the three lowest inflow discharges. As initial condition the weir's upstream side of the channel was filled with water to the weir's height. The fact that these three profiles are lower than the initial condition is due to a problem with FullSWOF_2D that lost water from the top boundary. These simulations were discarded for the continuation of the experiment. The height value for the remaining simulations was extracted 1.2 m before the weir base, to avoid observations in the acceleration zone, happening in proximity of the weir crest. At this point a space average over the channel breadth was taken. The procedure was repeated for all of the experiments.

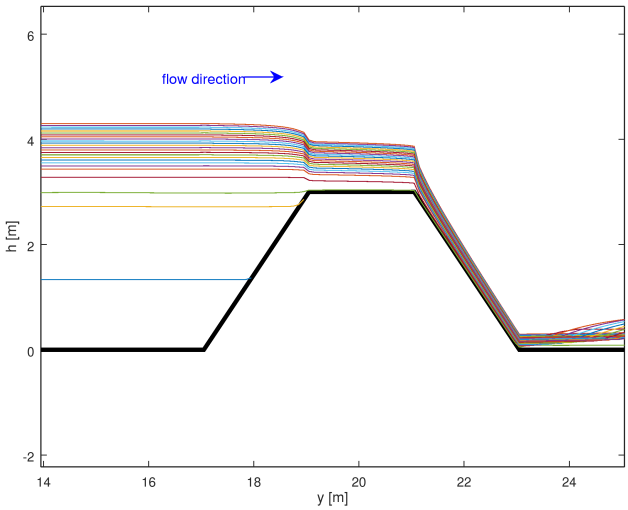


FIGURE 3.4: Free surface profiles along the channel axis for the 25 experiments.

Fitting the data

The weir equation (Eq. 3.1) was fitted to the dataset previously extracted using linear regression. As the original weir equation is non-linear with respect to the parameters, it was thus linearized by taking the logarithm:

$$\log(Q) = \log(C) + \log(L) + a \cdot \log(h_w) \tag{3.2}$$

The values of the parameters a and C were determined from the linearized model. The performance of the fitted weir equation was compared with the relationship obtained using linear and cubic spline interpolation.

Computing the error

To evaluate the performance of the three models, an increasing number of observations were progressively removed, the models trained on the remaining observations

and evaluated on the leaved-out observations. For an amount k of observations removed, all possible combinations were tested. The number of model evaluations follows:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \tag{3.3}$$

Since this number grows very fast a subset of the initial dataset was used. This can be found in Tab. A.1 in the Appendix. The first and last points of the dataset were kept during all the *cross-validation* experiment in order to avoid wild extrapolation at the boundaries of the inputs space. From the remaining 12 points all possible combinations of 1 to 10 points were removed. The average root mean squared error (RMSE) observed for the removal of 1 up to 10 points is displayed in Fig .3.7.

3.1.3 Results and Discussion

In Fig. 3.5 the (Q, h_w) pairs extracted from the simulations are plotted. The gap in the lower part of the plot is due to the data that had to be discarded. The displayed data points represent the input-output set which was used to fit the weir equation and to apply the linear and cubic spline interpolation.

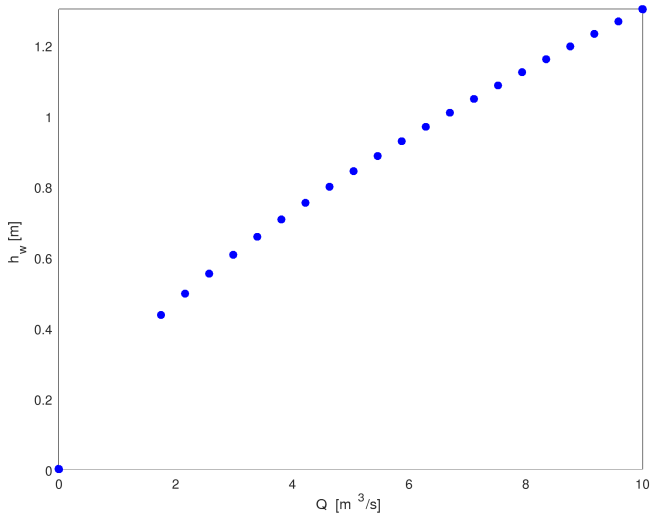


FIGURE 3.5: Plot of the (Q, h_w) pairs extracted from the simulations. Points corresponding to unsuccessful simulations were discarded, causing the gap visible in the bottom left part of the plot.

The fit of the different models to the input-output sets is displayed in Fig. 3.6. It can be observed that the linear interpolation fails to model the initial curvature which is instead captured by the fitting of the weir equation and the cubic spline interpolation. This is due to two main reasons. First, the density of data points in this region is low—the linear interpolation simply joins the two nearest data points with a segment—and on such a long distance it can diverge considerably from non-linear models. Secondly, this region of the input space presents a pronounced curvature, that makes the linear interpolation diverge faster.

With the remaining data points, all models seem to perform well: the three lines are almost perfectly overlaying, meaning that all give very similar results. Nevertheless, slight divergences can be observed in proximity of the data points, especially

between the weir equation and the two interpolation curves. This is due to the fact that the two interpolation models *interpolate* exactly the observations, whereas the curve obtained by fitting the weir equation (which represents the best linear fit to the observations minimizing the least-squares-error), does not necessarily pass exactly through the data points.

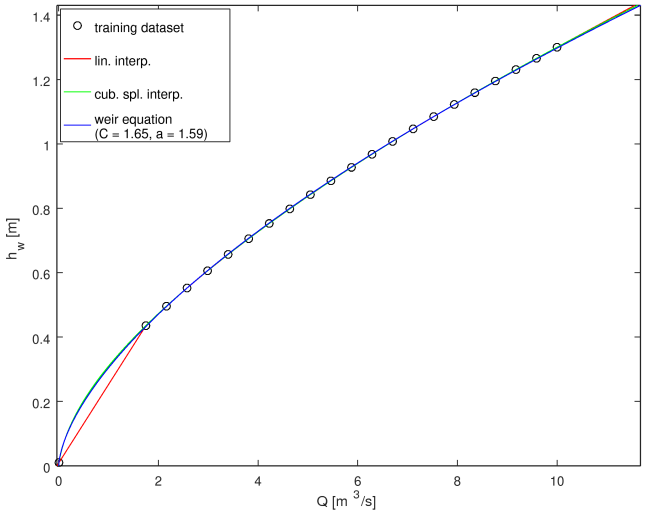


FIGURE 3.6: Fit of the three models to the simulated dataset. In blue the weir equation with $C = 1.65$ and $a = 1.59$, in red the linear interpolation through the observations and in green the cubic spline interpolation.

For the fitting of the weir equation the values $C = 1.65$ and $a = 1.59$ were found. As mentioned in Sec. 3.1.1, for a weir of 2 m breadth and vertical walls, C-values in the range $[1.36, 1.53]$ are expected. The value obtained for C assumes that C does not vary with h_w. Moreover, the weir used for the experiment has a trapezoidal cross section instead of a rectangular one. Tracy, 1957 investigated how the coefficient C varies with different weir shapes. For a weir with sloping walls the C coefficient increases, because the head losses are lower in comparison to one with vertical walls. The difference in the C coefficient observed here can be explained by this phenomenon.

Fig. 3.7 displays the RMSE obtained by cross-validation as explained in Sec. 3.1.2. No matter how many observations are used, the linear interpolation always performs worst. By increasing the number of observations its performance improves quite rapidly, but not enough to reach that of the other two models, because the process generating the observations does not have a linear behaviour.

The results for the cubic splines interpolation confirm what was already observed in Fig. 3.6: cubic splines interpolation seems to mimic in an accurate manner the process originating the observations. This is explainable by the high flexibility of cubic splines interpolation method, which typically shows very small errors when fitted with many observations. However, by reducing the number of observations available for interpolation, the error of this model increases exponentially. Despite this, on average if more than 5 observations are available, cubic splines interpolation outperforms the other methods.

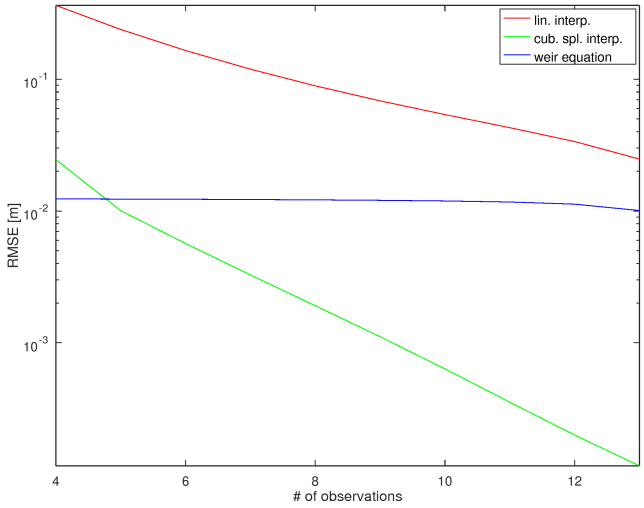


FIGURE 3.7: Mean RMSE of the three different models as a function of the number of observations used to train the models. It can be observed, that the RMSE of the cubic spline decreases exponentially by increasing the number of observations

On the other hand, the regression line obtained using the weir equation model shows only a slight decrease in accuracy when progressively removing observations for fitting. This high stability (insensitivity to the number of available observations for fitting) is due to the fact that in this case only two regression coefficients must be determined. Nevertheless, the low flexibility of linear regression models tends to prevent the interpolation of the data points and causes the linear regression model not to outperform cubic spline interpolation when many observations are available.

Fig. A.3 in the Appendix summarizes the performance of the three models when using $[4, 13]$ number of observations. Here it can be seen that the RMSE for the weir equation is higher on average respect to the cubic spline interpolation but still quite low (in the order of ≈ 1 cm). Its variation is almost inexistent; the box plot is practically described by a line.

An in-depth comparison between the two best models (see Fig. 3.8) shows new interesting features. The cubic spline interpolation is very sensitive to which observations are removed when few observations are available for interpolation. The standard deviation on its RMSE by changing which observations are removed increases by a factor of 60 when 4 resp. 13 observations are used. Oppositely, the regression line based on the weir equation model is more sensitive to which observation is removed when only few of them are removed; while it becomes less sensitive as fewer observations are used to fit the model. However, the variation in RMSE variability when 4 resp. 13 observations are used, changes only by a factor of ≈ 6.

When considering this, the regression line based on the weir equation can still have the potential to perform better than the cubic spline interpolation when only up to 6 observations are available (and not just by 4 as it was indicated by Fig. 3.7)

In the weir equation model proposed by Brown et al., 2009, the coefficient C was allowed to vary as a function of h_w: the shorter the weir width, the wider becomes the range of values for the C coefficient. If the regression line would have been fitted

on this model, then a better fit to the observations would have been obtained.

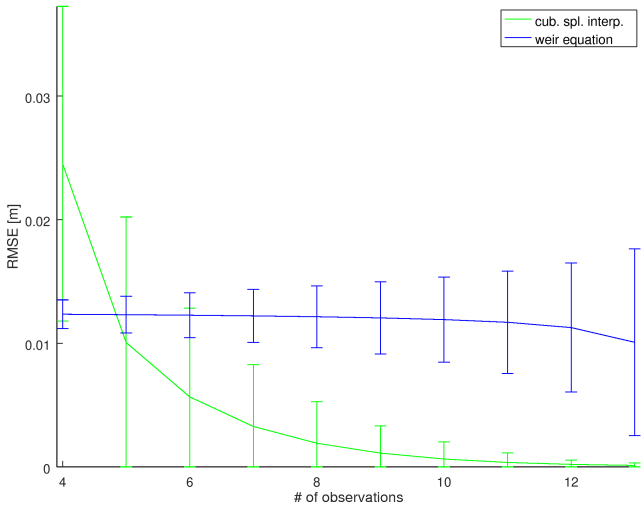


FIGURE 3.8: Closer comparison of the weir equation and the cubic spline interpolation models.

This first case studies highlighted the importance of the trade-off between *prior knowledge* and *data availability*. Machine learning (ML) regression algorithms such as *Random Forest* (Breiman, 2001), *Gradient Boosting Machines* (Friedman, 2001) and *Feedforward Neural Networks* (James et al., 2013) are known to be very powerful at establishing non-linear functional relationship hidden in the data, but they also are particularly data-greedy. They have the potential to perform very well when a big training dataset is available, but fail if not enough data are available. Using numerical simulators, the generation of these large training datasets can be computationally very costly, thus partially hampering the use of these algorithms.

In this case study, it was shown that simpler interpolation methods can still allow to obtain good performance when moderate simulated input-output sets are available. Moreover, the encoding of prior knowledge was proved to be useful when less data are available: the *weir equation* model, unlike the others, could still perform well when fitted to few observations.

Taking into consideration the previous findings, GPs could be the ideal tool for emulation of physical processes due to their ability to model highly non-linear relationships, to interpolate the data points as well as encoding prior knowledge of the process (Rasmussen and Nickisch, 2010). Recent studies showed the possibility to direct map partial differential equations (PDEs) into covariance functions (Lindgren, Rue, and Lindström, 2011). This allows to include the mechanistic of the process in the interpolation problem. On the other side, identification of accurate GP models can shed new light on the mechanistic of the process.

Case study 2

3.2 A prelude to an early flood warning system

As already stated in the introduction, our society and its infrastructures are exposed to the risk of flooding. In this case study, we illustrate a methodology which can be used to develop an emulation-based early flood warning system. For a channel conveying water from a catchment, we want to estimate, at a specific location along the channel, the time needed to reach a specific *threshold discharge* ($Q_!$). The time needed to exceed $Q_!$ will be referred hereinafter as *time-to-threshold* ($t_!$). This time-to-threshold will depend on the rainfall intensity on the catchment and the initial soil moisture content.

With the term "rain event", we are going to consider a combination of the *rain intensity* and the *initial soil saturation* over the catchment. For some rain events, namely very low precipitation and low initial soil saturation conditions, the threshold discharge at the specific location may never be reached. For this reason, the developed warning system has a hierarchical structure: it is composed of a classifier determining whether the threshold discharge ($Q_!$) will be exceeded and of an emulator which estimates the time-to-threshold $t_!$.

To develop the methodology, a synthetic catchment topography was generated. Due to the generalization and abstraction of the proposed work-flow however, the methodology can be easily applied to real-situations.

3.2.1 Material and methods

Generating the topography

The synthetic topography used is displayed in Fig. 3.9 and covers a catchment of size $2\,\text{km} \times 2\,\text{km}$. The simulation grid is composed of 100×100 cells, which results in a cell resolution of $20\,\text{m} \times 20\,\text{m}$.

The topography is generated by the superposition of a sloping plane with three Gaussian bumps. The Gaussian bumps have different heights and widths and generate a *Y-shaped channel* which extends from the upper and left boundary down to the lower boundary. This has a single outlet located close to the center of the domain's bottom boundary.

In contrast to a real topography, the synthetic topography has a much smoother surface. This allows the use of a coarse grid resolution without losing the topographical features and the reduction of the simulation runtime. In a real situation, a wigglier topography might introduce additional non-linearities in the runoff-generation process, which would require an higher resolution of the simulation grid and thus longer computing time.

Setting-up the simulations

The dataset required for building the emulator was generated by running 50 simulations with different combinations of the variables *rain intensity* (I) and *initial soil saturation* (θ_i). The duration of every simulation was $\approx 30\,\text{min}$, for a simulated time of 9 h, giving a ratio real-time/simulation-time of $3.3\,\text{min}\,\text{h}^{-1}$ simulated. The initial soil saturation, despite being a spatially distributed variable, was kept uniform over the whole domain. The rain intensity was kept constant and was uniformly

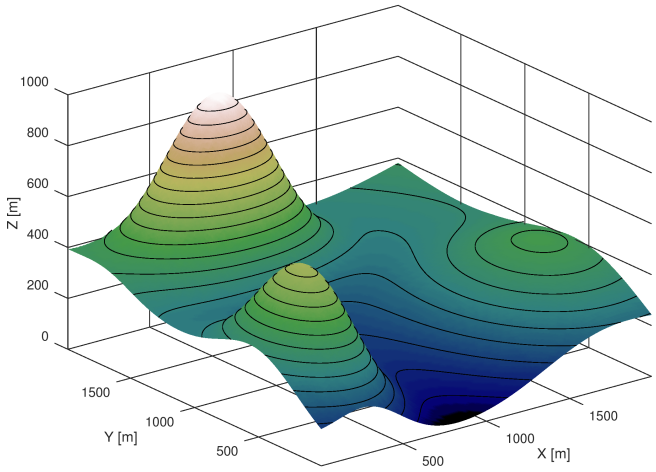


FIGURE 3.9: Synthetic topography composed of three Gaussian bumps on a sloping plane. The runoff from this domain, due to the high variability of its slope, is given by the superpositions of different waves. For the time-to-threshold flood-warning problem, this generates non-monotonous functions which make emulation challenging (see Fig. 3.10).

applied over the domain, also because FullSWOF_2D, at this stage of development, only allows for uniformly distributed rain (Laguerre, 2016).

5 different initial saturations in the $[0, 1]$ interval and 10 rain intensities in the $[10, 35]\,\mathrm{mm\,h^{-1}}$ interval were taken and all their possible combinations were used as inputs for the simulations. The rain duration was set to 6 h, while a simulation duration of 9 h was chosen in order to be able to observe the hydrograph recession. A time resolution of 60 s was used.

Tab. 3.2 reports other simulation parameters that were kept constant over all the simulations. The parameters marked with * are those *spatially distributed*, meaning that a different value could be set for every cell. For simplicity, a uniform catchment was used, therefore the values from the table are valid for the whole domain. For the bottom boundary a *Neumann boundary condition* was selected, allowing water to freely outflow. Three *wall boundary conditions* were set for the remaining boundaries, making sure that all of the water is lost through the bottom one.

Extracting the datasets

The datasets used to build the emulator was extracted from the simulation outputs in two steps. First, the outflow hydrographs were extracted from the simulation output files, by summing up, at every time-step saved, the cell discharge along the whole domain bottom boundary. The result of this operation is a discharge time series with 540 observations, with a temporal resolution of 1 minute. The same procedure was repeated for each of the 50 events, producing 50 different hydrographs (see Fig. 3.11).

Successively, the time-to-threshold $t_!$ was computed based on the specified threshold discharge $Q_!$. Simulated $t_!$, paired with the corresponding rain intensity and the initial soil saturation condition, formed the output-inputs set of observations generating the *training dataset* used to construct the emulator.

TABLE 3.2: Parameters and setting fixed for all simulations. Parameters marked with * are spatially distributed. For simplicity, these were kept uniform for the whole domain.

Parameter	**Value**	**Units**
Domain x-length	2000	m
Domain y-length	2000	m
Number of cells x	100	
Number of cells y	100	
Friction coefficient*	0.03	$s\,m^{-1/3}$
Crust thickness*	1	m
Crust hydraulic conductivity*	$2 \cdot 10^{-6}$	$m\,s^{-1}$
Soil hydraulic conductivity*	$2 \cdot 10^{-6}$	$m\,s^{-1}$
Soil suction head*	0.09	m
Soil maximum infiltration rate*	19.8	$mm\,h^{-1}$

In a real situations, $Q_!$ could be the discharge at which a bridge or the embankments located on the channel are overflown. In order to develop the methodology, $Q_!$ was set to 10 % of the Q_{max} value recorded, corresponding to $0.17\,m^3\,s^{-1}$.

By looking at one of the hydrographs extracted (Fig. 3.10) it can be observed that $t_!$ is a difficult quantity to predict. In fact, depending on the $Q_!$ set, the $t_!$ can be reached very quickly, can show a long time-lag, or not be reached at all. As illustratory example, in Fig. 3.10, the hydrograph between $Q_!$ and the $Q'_!$ presents a plateau. Thus, a minimal variation of the threshold discharge specification can lead to sudden large jump of the time-to-threshold. For this reason, the studied quantity $t_!$ is said to be *discontinuous*.

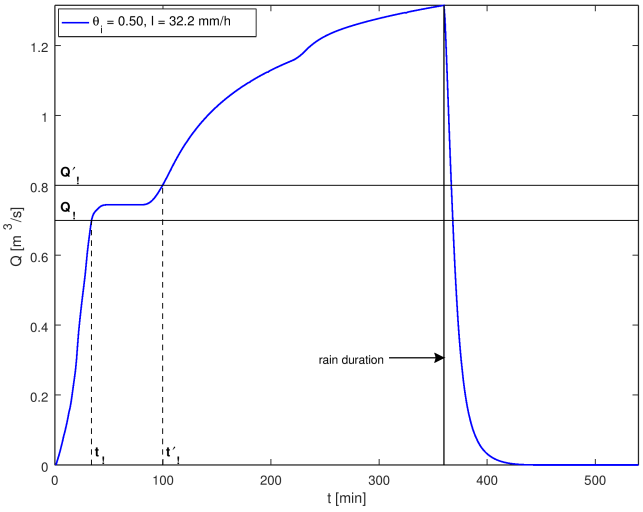


FIGURE 3.10: Response hydrograph for $\theta_i = 0.5$ and $I = 32.2\,mm\,h^{-1}$. By looking at the plateau located between the $Q_!$ and $Q'_!$ lines, the discontinuity in $t_!$ can be observed. A minimal change in the threshold discharge causes a jump of the time-to-threshold of almost an hour. This feature makes the emulation task very challenging.

Building-up the classifier

The classifier was built in order to ease the task of computing the time-to-threshold. The classifier should distinguish whether a rain event is going to reach the threshold discharge or not. Only the rain events classified as "reaching the discharge threshold" are given to the emulator to compute *when* this will happen.

To train the classifier we assigned the value -1 to rain events not reaching the discharge threshold and 1 the others. We tested different GP classifier before finding a satisfying one. Since the data points seemed to be linearly separable, we decided to use the "meanLinear" mean function and vary the covariance functions. To visually inspect the performance of each of these different classifiers, we drew the classifier boundaries over the input space formed by the rain intensities and the initial soil saturation conditions. Fig. A.4 shows the classifier boundary obtained by using a "covConst" covariance function.

However, an infinite number of curves could define the frontier between rain events exceeding and not the threshold discharge. Thus we decided to perform new simulations using combinations of rainfall intensity and initial soil saturation condition located in proximity of the classifier boundary.

The data points obtained by these new simulations (marked as triangle in 3.12) suggested that the boundary was actually not linear but curvy. We therefore trained various GP classifiers by testing various curvy mean function and a set of covariance functions.

Building-up the emulator

The procedure we used to build the time-to-threshold emulator is similar to the one used to build up the classifier. To select an accurate GP model and evaluate its performance, we performed other simulations with FullSWOF_2D to also create a :

1. *validation dataset* : composed of 36 different (I, θ_i) combinations covering regularly the regions of the input space not covered by the training data. These data points are displayed in blue in Fig. 3.13 , while the code found to generate this validation observations is provided in Sec. A.3.1 of the Appendix.

2. *test dataset* : composed of 6 (I, θ_i) observations located sparsely in the inputs space. These observations are displayed in green in Fig. 3.13, and the (I, θ_i) pairs values are reported in Tab. A.2 of the Appendix.

To define an accurate GP model, we tested various combinations of mean and covariance functions. The GP models trained on the training dataset (composed of 50 observations) were evaluated on the validation dataset. The mean and covariance functions of the GP model showing the lowest RMSE (computed on the validation set) was selected to build the final emulator.

In order to increase the details captured by the final emulator, the training and validation datasets were combined together, and the resulting set of observations used to train the final GP model.

The performance of the final emulator was assessed on the test dataset, by computing the RMSE and the maximum absolute error (MAE).

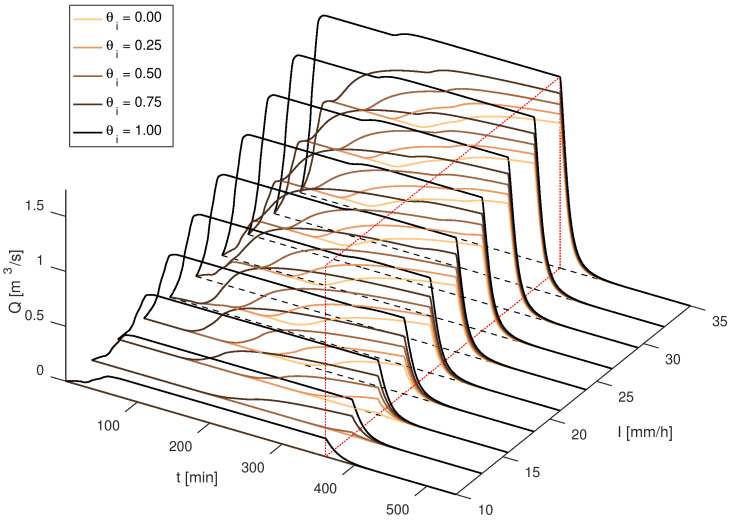


FIGURE 3.11: Response hydrographs for the 50 simulations at the catchment outlet which compose the training dataset. The red frame shows the end of the rain event. Depth-dimension shows the *rain intensity* variable, while the *initial saturation* variable is rendered by the color map. It can be observed, that shape and magnitude of the response vary considerably from one simulation to the other.

3.2.2 Results

Simulations

Fig. 3.11 shows the 50 simulated hydrographs which compose the training dataset. Many interesting features can be observed. First of all, it can be seen that experiments run with the two lowest rain intensities and low initial soil saturation generated no discharge. In the second place, simulations run with $\theta_i = 1$ always reached their peak discharge, and this happened quite quickly. Some of these show a second smaller increase after the first quick rising limb. This is even more accentuated in the simulations were the soil was initially not saturated. We can explain this effect as the mapping of the topography configuration to the response hydrographs. The topography used presents zones with very steep slopes and others with very flat ones, causing different flow velocities. When the water from the inclines has reached the outlet, that coming from flat areas is still traveling downstream. After a given lapse of time this water reaches the outlet as well, giving rise to the bumps visible in the hydrographs.

The classifier

Fig. 3.12 displays the results of the classification task. Rain events reaching the discharge threshold are marked with red circles, while rain events not reaching it with green ones. Circles represent training dataset observations, whereas triangles indicate the dataset which was added in a second time to test the linearity of the boundary. According to the classifier, every (I, θ_i) pair chosen in the yellow region will cause the exceeding of $Q_!$. The classifier could correctly classify all but one data points given.

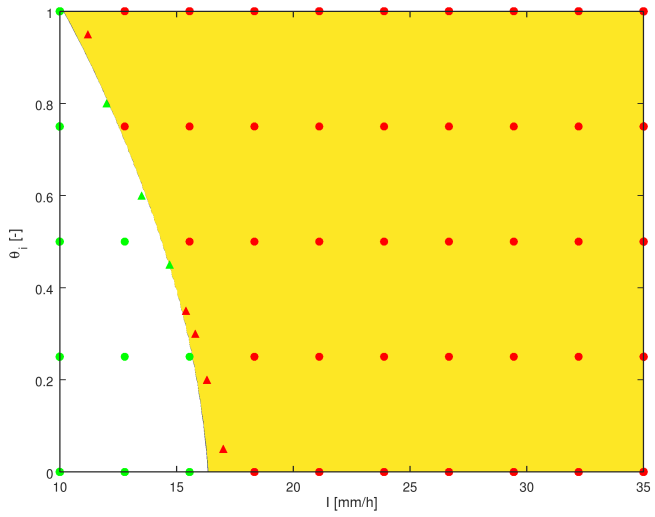


FIGURE 3.12: Classifier: the circles represent the first training dataset, while the triangles the second one. In red all events exceeding the threshold, in green all those which did not. Events in the yellow region are the ones classified as "exceeding the threshold", for which the time-to-threshold has to be estimated.

The emulator

The time-to-threshold emulator is shown in Fig. 3.13. The x and y axis indicate the value of the inputs (rain intensity and soil saturations) while the corresponding time-to-threshold is given on the z-axis. The circles represent the datasets used for training (red), validation (blue) and testing (green). The interpolation surface of the final emulator is given by the black mesh and is the result of the evaluation of the final GP model at 6400 mesh points.

The performance of the emulator was assessed by computing MAE and RMSE on the test dataset. These provides an estimation of the out-of-sample error, and is reported in Tab. 3.3.

TABLE 3.3: Emulator performance on the *test* datasets.

	MAE [min, %]	**RMSE [min]**
test	6.0, 5.9	4.0

The emulator's speedup factor was also computed. The evaluation of the emulator at one point lasts in average 0.012 s, whereas running a simulation takes approximately 30 min. This represents a *speedup* of **150 000** $\times$.

3.2.3 Discussion

In early flood warning system, a crucial point is not to underestimate the danger of floods. In case a flood alert is issued but no flood occurs, minor economic loss could be related to deployment of temporary flood control measures or the cost of population evacuations, but nobody would be armed. In the opposite scenario, when no alert is issued but a flood occurs, the risk of elevated economic losses and fatalities occurrence is high.

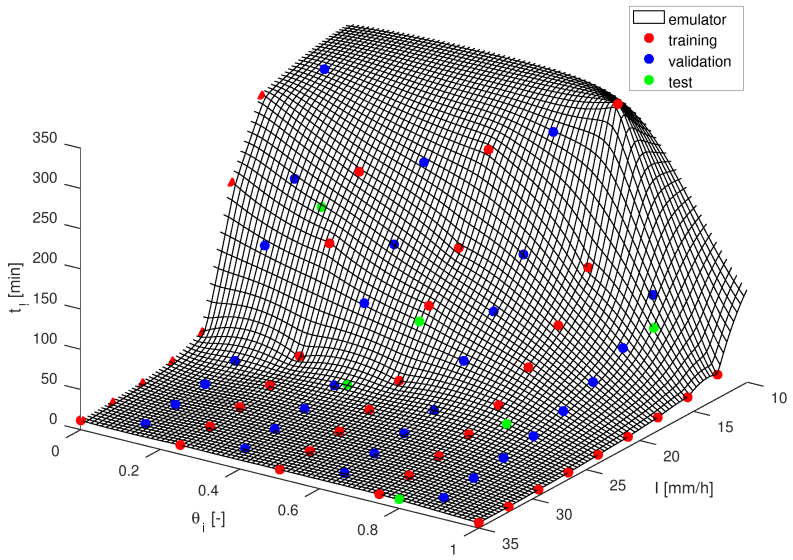


FIGURE 3.13: Time-to-threshold emulator: training (red), validation (blue) and test (green) datasets and the emulator (mesh) predicting the time-to-threshold ($t_!$) in regions where there are no observations. The emulator interpolates all of the training observations (red and blue ones). The extrapolation far outside the observations boundaries are meaningless. These are already discarded by the classifier.

Our emulation-based early flood warning system is composed of a *classifier* which says if an alert should be emitted, and by an *emulator* which estimates the time period before a specific discharge is exceeded in case an alert is emitted by the classifier.

Dramatical consequences could thus happen if the classifier would not emit an alert, but the specified discharge is exceeded.

In Sec. 3.2.2 we saw that the proposed classifier failed to classify one observation. In that case, the classifier predicted that the discharge threshold would have been exceeded when actually was not. However the misclassified point is a training point, which means that the established classifier is not so accurate and a better one should be identified. To do this, new simulations near the actual classifier boundary should be performed , new GP models tested and evaluated. This iterative procedure has not be done due to lack of time, but should be conducted if such type of emulator would be applied in a real situation.

As far as the estimation of time-to-threshold is concerned, underestimation of the period of time before the threshold discharge is reached has less consequences than an overestimation of it. This because the flood mitigation measures (for the expected flood) would already be in place. Tab. A.4 in the Appendix reports the errors in time-to-threshold for each observation of the test set.

Regarding underestimation of $t_!$, the worst error in term of magnitude on the test set is when estimating the flood to occur after 198.0 min when actually occurring 5.5 min before, based on the numerical simulations of test observation 2. This means a relative time-to-threshold error of about 3 % to the simulated time-to-threshold.

However, when having to put in place flood mitigation measures, high accuracy in determining the time-to-threshold is required when the time period is short. In

the case of the test observations 4 and 5, the relative time-to-threshold error is of about 4.5 %.

Although having obtained a time-to-threshold error of 30 s when $t_!$ is in the order of 10 min is an excellent result, more test observations should be simulated to obtain a reliable emulator performance assessment.

As a final remark regarding the emulator performance, it should be kept in mind that the time resolution of the numerical simulation was 1 min. Thus, the test time-to-threshold error with magnitude smaller than 1 min should not be considered de facto as errors. This is the case for test observations 4 and 6.

In light of real-world applications, the results obtained with our emulator are over-optimistic. The actual emulator performance does not account for the uncertainty in the inputs parameters. In real situations, rain intensity and initial soil saturation must be estimated or are known only to a given degree of accuracy. Therefore, propagation of inputs uncertainty to emulator outputs should be accounted when assessing the performance of an operational emulator.

Still, the advantage in using emulators is that uncertainty quantification can be performed much faster than using numerical simulators: the speed up obtained with our emulator was shown to be 150 000 $\times$.

In conclusion, the shorter computing time required by the emulator would also allow to adapt the predictions in near-real time: the emulator could be re-evaluated during the duration of the rain event if more accurate inputs conditions are available or integrating for example in the workflow a Kalman filtering data assimilation scheme.

Chapter 4

Conclusions

This thesis follows a storyline, which connects up the dots, resulting in a prelude to emulation for near-time flood prediction. Venturing in a completely new field, that of emulation, opened me up a whole new world. This itinerary through emulation, after clearing up some initial doubts, made me realize its true potential. Emulation can have great applicability, especially in the engineering fields. Engineering often has to tackle repetitive problems, which solution can be found only iteratively. Emulation can provide alternative new ways to solve such problems. Once an appropriate emulator for a specific task is built, this can provide fast answer to the problem of interest. Emulation can be performed at almost no cost: it requires neither special tools nor special investments. The work done on this thesis demonstrate it.

In spite of its simplicity, the first case study was for me a very didactic one. The weir equation, commonly used in hydraulic engineering, proved to be a fair approximation of the relation discharge-height over the weir, although flexible non parametric interpolation techniques have shown to produce more accurate estimates when an adequate number of observations is available. This case study also shows the potential of numerical simulation in replacing the realization of laboratory experiments, with the potential of reducing future research costs and time.

Although the methodology in the second case study appears more complex than the one adopted in the first one, the high level of abstraction and generalization of the presented workflow allows to apply the proposed early flood warning system to real situations with only slight modifications. These especially concerns the assessment of the emulator performance. Still, the partial accuracy assessment conducted reported excellent performance, especially with regards to the estimation of the time period before a flood occurs.

In conclusion, this thesis shows the benefit of conjugating the accuracy of numerical simulation with the huge enhancement in computing time provided by emulators. In the engineering field, promising research areas may include uncertainty propagation, sensitivity analysis and models optimization or calibration, which are actually hampered by the excessive computational burden of numerical simulation.

Bibliography

Alfieri, L. et al. (Jan. 23, 2015). "Global warming increases the frequency of river floods in Europe". In: *Hydrology and Earth System Sciences Discussions* 12.1, pp. 1119–1152. ISSN: 1812-2116. DOI: 10.5194/hessd-12-1119-2015. URL: http://www.hydrol-earth-syst-sci-discuss.net/12/1119/2015/ (cit. on p. 1).

Ali, Mohamed Sukri Mat, Con J. Doolan, and Vincent Wheatley (2009). "Grid convergence study for a two-dimensional simulation of flow around a square cylinder at a low Reynolds number". In: *Seventh International Conference on CFD in The Minerals and Process Industries (ed. PJ Witt & MP Schwarz)*, pp. 1–6 (cit. on p. 11).

Berghuijs, W. R., R. A. Woods, and M. Hrachowitz (July 2014). "A precipitation shift from snow towards rain leads to a decrease in streamflow". In: *Nature Climate Change* 4.7, pp. 583–586. ISSN: 1758-678X, 1758-6798. DOI: 10.1038/nclimate2246. URL: http://www.nature.com/articles/nclimate2246 (cit. on p. 1).

Blöschl, Günter et al. (Aug. 11, 2017). "Changing climate shifts timing of European floods". In: *Science* 357.6351, pp. 588–590. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aan2506. URL: http://www.sciencemag.org/lookup/doi/10.1126/science.aan2506 (cit. on p. 1).

Bos, M. G. (1989). *Discharge measurement structures*. Third revised edition. International Institute for Land Reclamation and Improvement (ILRI), Wageningen, The Netherlands. ISBN: 90 70754 15 0 (cit. on p. 9).

Breiman, Leo (Oct. 1, 2001). "Random Forests". In: *Machine Learning* 45.1, pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324 (cit. on p. 17).

Brown, S. A. et al. (Sept. 2009). *Urban Drainage Design Manual*. FHWA -NHI- 10- 009 HEC-22. Federal Highway Administration, p. 478 (cit. on pp. 10, 16).

Carbajal, J. P., João Paulo Leitão, et al. (2016). "Appraisal of data-driven and mechanistic emulators of nonlinear hydrodynamic urban drainage simulators". In: *arXiv preprint arXiv:1609.08395* (cit. on pp. 1, 3).

Carbajal, J. P. and J. Rieckermann (2017). *EmuMore Project website*. EmuMore's blog. URL: https://kakila.bitbucket.io/emumore/about (visited on 03/11/2018) (cit. on pp. 3, 4).

Christensen, O.B. and J.H. Christensen (Dec. 2004). "Intensification of extreme European summer precipitation in a warmer climate". In: *Global and Planetary Change* 44.1, pp. 107–117. ISSN: 09218181. DOI: 10.1016/j.gloplacha.2004.06.013. URL: http://linkinghub.elsevier.com/retrieve/pii/S0921818104001006 (cit. on p. 1).

community, Inkscape (2018). *Inkscape*. Version 0.92. URL: www.inkscape.org (visited on 03/21/2018) (cit. on p. vii).

Dankers, Rutger and Luc Feyen (Aug. 27, 2009). "Flood hazard in Europe in an ensemble of regional climate scenarios". In: *Journal of Geophysical Research* 114 (D16). ISSN: 0148-0227. DOI: 10.1029/2008JD011523. URL: http://doi.wiley.com/10.1029/2008JD011523 (cit. on p. 1).

Delestre, Olivier et al. (2017). "FullSWOF: Full Shallow-Water equations for Overland Flow". In: *The Journal of Open Source Software* 2.20, p. 448. DOI: 10.21105/

joss.00448. URL: https://www.idpoisson.fr/fullswof/ (cit. on pp. vii, 2, 6, 11, 13, 19, 21, 37).

Eaton, John W. et al. (2016). *GNU Octave version 4.2.0 manual: a high-level interactive language for numerical computations*. URL: http://www.gnu.org/software/octave/doc/interpreter (cit. on p. 38).

European Environment Agency (May 6, 2013). *Flood risk in Europe: the long-term outlook*. Copenhagen, Denmark: European Environment Agency (EEA), p. 8. URL: https://www.eea.europa.eu/highlights/flood-risk-in-europe-2013 (cit. on p. 1).

Friedman, Jerome H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine". In: *The Annals of Statistics* 29.5, pp. 1189–1232. ISSN: 00905364. URL: http://www.jstor.org/stable/2699986 (cit. on p. 17).

Hall, J. et al. (June 30, 2014). "Understanding flood regime changes in Europe: a state of the art assessment". In: *Hydrology and Earth System Sciences Discussions* 10.12, pp. 15525–15624. ISSN: 1812-2116. DOI: 10.5194/hessd-10-15525-2013. URL: http://www.hydrol-earth-syst-sci-discuss.net/10/15525/2013/ (cit. on p. 1).

Hirabayashi, Yukiko, Shinjiro Kanae, et al. (Aug. 2008). "Global projections of changing risks of floods and droughts in a changing climate". In: *Hydrological Sciences Journal* 53.4, pp. 754–772. ISSN: 0262-6667, 2150-3435. DOI: 10.1623/hysj.53.4.754. URL: https://doi.org/10.1623/hysj.53.4.754 (cit. on p. 1).

Hirabayashi, Yukiko, Roobavannan Mahendran, et al. (June 9, 2013). "Global flood risk under climate change". In: *Nature Climate Change* 3.9, pp. 816–821. ISSN: 1758-678X, 1758-6798. DOI: 10.1038/nclimate1911. URL: http://www.nature.com/doifinder/10.1038/nclimate1911 (cit. on p. 1).

James, Gareth et al. (2013). *An Introduction to Statistical Learning*. Vol. 103. Springer Texts in Statistics. New York, NY: Springer New York. ISBN: 978-1-4614-7137-0 978-1-4614-7138-7. DOI: 10.1007/978-1-4614-7138-7. URL: http://link.springer.com/10.1007/978-1-4614-7138-7 (visited on 11/03/2017) (cit. on p. 17).

Köplin, N. et al. (Feb. 15, 2014). "Seasonality and magnitude of floods in Switzerland under future climate change". In: *Hydrological Processes* 28, pp. 2567–2578. ISSN: 08856087. DOI: 10.1002/hyp.9757. URL: http://doi.wiley.com/10.1002/hyp.9757 (cit. on p. 1).

Laguerre, Christian (Mar. 14, 2016). *Documentation of FullSWOF_2D v1.07.00 (2016-03-14)* (cit. on p. 19).

Lindgren, Finn, Håvard Rue, and Johan Lindström (Sept. 2011). "An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach: Link between Gaussian Fields and Gaussian Markov Random Fields". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.4, pp. 423–498. ISSN: 13697412. DOI: 10.1111/j.1467-9868.2011.00777.x. URL: http://doi.wiley.com/10.1111/j.1467-9868.2011.00777.x (cit. on p. 17).

Milly, P. C. D. et al. (Jan. 31, 2002). "Increasing risk of great floods in a changing climate". In: *Nature* 415, pp. 514–517 (cit. on p. 1).

Octave community (2018). *GNU Octave*. Version 4.2.1. URL: www.octave.org (visited on 03/21/2018) (cit. on pp. vii, 2, 5, 7, 10, 37).

Rasmussen, Carl Edward and Hannes Nickisch (2010). "Gaussian processes for machine learning (GPML) toolbox". In: *The Journal of Machine Learning Research* 11, pp. 3011–3015. URL: http://www.gaussianprocess.org/gpml (cit. on pp. 5, 17).

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. OCLC: ocm61285753. Cambridge, Mass: MIT Press. 248 pp. ISBN: 978-0-262-18253-9 (cit. on p. 5).

Stahl, K. et al. (Feb. 14, 2012). "Filling the white space on maps of European runoff trends: estimates from a multi-model ensemble". In: *Hydrology and Earth System Sciences Discussions* 9.2, pp. 2005–2032. ISSN: 1812-2116. DOI: 10.5194/hessd-9-2005-2012. URL: http://www.hydrol-earth-syst-sci-discuss.net/9/2005/2012/ (cit. on p. 1).

The FullSWOF Team (2018). *The FullSWOF Project*. Institut Denis Poisson. URL: https://www.idpoisson.fr/fullswof/ (visited on 02/13/2018) (cit. on p. 6).

Thober, Stephan et al. (Jan. 1, 2018). "Multi-model ensemble projections of European river floods and high flows at 1.5, 2, and 3 degrees global warming". In: *Environmental Research Letters* 13.1, p. 014003. ISSN: 1748-9326. DOI: 10.1088/1748-9326/aa9e35. URL: http://stacks.iop.org/1748-9326/13/i=1/a=014003?key=crossref.c93f4ad828d9152aa572057fa0c67675 (cit. on p. 1).

Tracy, H. J. (1957). *Discharge characteristics of borad-crested weirs*. 397. Washington, D. C.: United States Geological Survey (cit. on p. 15).

Walcott, Charles D. (1907). "Weir experiments, coefficients and formulas". In: p. 235 (cit. on p. 9).

Westra, Seth, Lisa V. Alexander, and Francis W. Zwiers (June 2013). "Global Increasing Trends in Annual Maximum Daily Precipitation". In: *Journal of Climate* 26.11, pp. 3904–3918. ISSN: 0894-8755, 1520-0442. DOI: 10.1175/JCLI-D-12-00502.1. URL: http://journals.ametsoc.org/doi/abs/10.1175/JCLI-D-12-00502.1 (cit. on p. 1).

Zolina, Olga et al. (Mar. 2010). "Changing structure of European precipitation: Longer wet periods leading to more abundant rainfalls". In: *Geophysical Research Letters* 37.6, n/a–n/a. ISSN: 00948276. DOI: 10.1029/2010GL042468. URL: http://doi.wiley.com/10.1029/2010GL042468 (cit. on p. 1).

List of Abbreviations

EEA	**E**uropean **E**nvrionmental **A**gency
EP	**E**xpectation **P**ropagation
FEM	**F**inite **E**lement **M**ethod
FV	**F**inite **V**olume
FVM	**F**inite **V**olume **M**ethod
GCI	**G**rid **C**onvergence **I**ndex
GC	**G**aussian **P**rocess
HPC	**H**igh **P**erformance **C**omputing
IQR	**I**nter**Q**uartile **R**ange
LS	**L**east**S**quares
ML	**M**achine **L**earning
MAE	**M**aximum **A**bsolute **E**rror
RMSE	**R**oot **M**ean **S**quared **E**rror
SWE	**S**hallow **W**ater **E**quation
UQ	**U**ncertainty **Q**uantification

List of Symbols

dx	Grid spacing in x-direction	m
dy	Grid spacing in x-direction	m
Lx	Domain length in x-direction	m
Ly	Domain length in y-direction	m
N_{states}	Number of intermediate states saved	
Nx	Number of cells in x-direction	
Ny	Number of cells in y-direction	
Q	Discharge	$m^3\,s^{-1}$
Q_{max}	Maximum discharge	$m^3\,s^{-1}$
t_{max}	Simulation duration	s

Appendix A

Appendix

A.1 Additional material

The GNU Octave fswof2d package can be downloaded at:

- https://bitbucket.org/binello7/fswof2d/

All of the material composing these thesis (documents, scripts, files, functions, etc.) is available at:

- https://bitbucket.org/binello7/master_thesis

A.2 Case study 1

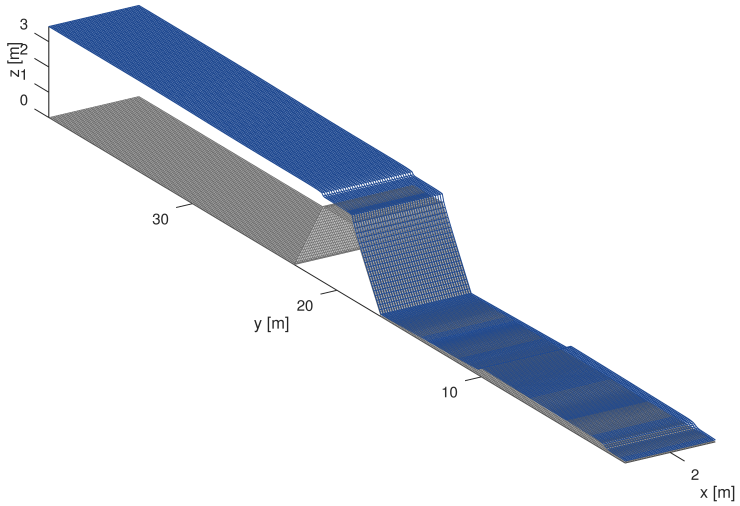


FIGURE A.1: Channel topography and free surface of the water computed with FullSWOF_2D at $t = 16\,\text{s}$. The simulation at this point has not reached the steady-state conditions yet. This is visible from the waves on the water surface downstream of the weir, indicating that the hydraulic jump has not stabilized yet.

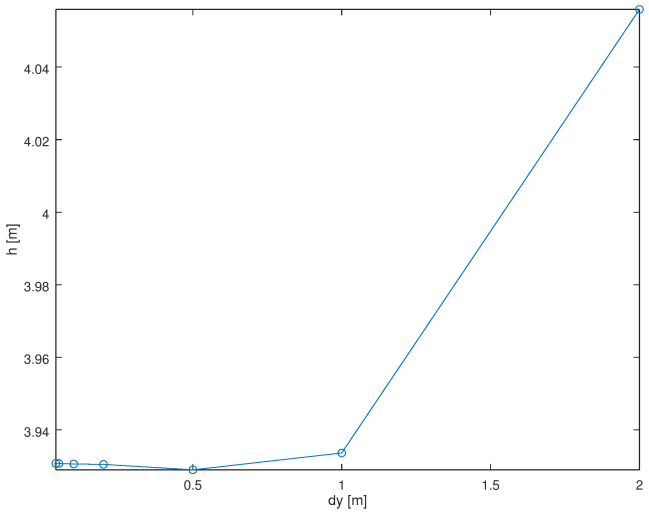


FIGURE A.2: Measured free surface height at the weir crest as a function of the grid resolution used.

TABLE A.1: Dataset used for computing the model error.

h_w [m]	0.00	0.50	0.55	0.61	0.66	0.80	0.89	0.93	1.05	1.08	1.20	1.16	1.27	1.30
Q [m^3 s^{-1}]	0.00	2.16	2.58	2.99	3.40	4.64	5.46	5.88	7.11	7.53	8.76	8.35	9.59	10.00

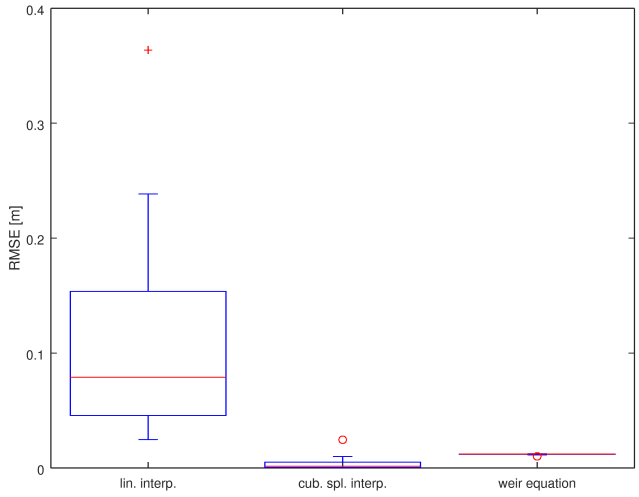


FIGURE A.3: Box plot summarizing the performance of the three models by using [4, 13] # of points. The red line represent the median RMSE. The box is given by the 1st and 3rd quartile. Whiskers are drawn at 1.5 $\times$ IQR. Points between 1.5 $\times$ IQR and 3 $\times$ IQR are marked with 'o', while '+' indicates points beyond this range (Eaton et al., 2016).

A.3 Case study 2

A.3.1 Validation dataset

```
## Test data points in the centre of the training dataset mesh
# rain intensity: 1st and last point location
maxri = 35; minri = 10; # max, min ri values
nri = 9; # n. ri values
ri1 = minri + ((maxri - minri) / nri) / 2; # ri value 1st point (center)
ri2 = maxri - ((maxri - minri) / nri) / 2; # ri value last point (center)

# all ri values
rain_intensities_test = [linspace(ri1, ri2, nri)].'; #[mm/h]

# initial soil saturation: 1st and last point location
maxss = 1; minss = 0; # max, min ss values
nss = 4; # n. ss values
ss1 = minss + ((maxss - minss) / nss) / 2; # ss value 1st point (center)
ss2 = maxss - ((maxss - minss) / nss) / 2; # ss value last point (center)

# all ss values
soil_saturations_test = [linspace(ss1, ss2, nss)].'; #[-]
```

A.3.2 Test dataset

TABLE A.2: Input parameters of the test set

	1	**2**	**3**	**4**	**5**	**6**
$\boldsymbol{I}$ [mm h^{-1}]	15	20	22	25	25	35
$\boldsymbol{\theta_i}$	0.9	0.2	0.5	0.8	0.4	0.8

A.3.3 Additional dataset for classification

TABLE A.3: Dataset used to refine the classification boundary.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
$\boldsymbol{I}$ [mm h^{-1}]	11.2	12.0	13.5	14.7	15.4	15.8	16.3	17.0
$\boldsymbol{\theta_i}$	0.95	0.80	0.60	0.45	0.35	0.3	0.20	0.05

A.3.4 Linear GP classifier

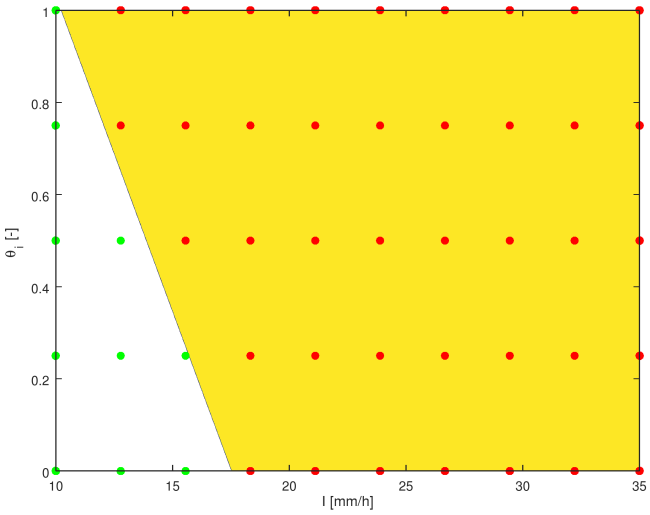


FIGURE A.4: Linear separation of the training dataset using GP with "meanLinear" and "covConst" functions.

A.3.5 Time-to-threshold test performance

TABLE A.4: Emulator's time-to-threshold errors on test observations. Error are computed by substracting emulated $t_!$ from simulated $t_!$. Negative errors indicate overstimation of the time-to-threshold.

	1	**2**	**3**	**4**	**5**	**6**
simulated $t_!$ [min]	91.5	192.5	101.5	32.5	32.5	11.5
emulated $t_!$ [min]	86.2	198.0	95.5	32.4	34.0	11.0
error ϵ [min]	5.3	-5.5	6.0	0.1	-1.5	0.5