Goeury, Cédric; Zaoui, Fabrice; Audouin, Yoann; Prodanovic, Pat; Fontaine, Jacques; Tassi, Pablo; Ata, Riadh

Finding Good Solutions to Telemac Optimization Problems with a Metaheuristic

Finding Good Solutions to Telemac Optimization Problems with a Metaheuristic

C. Goeury[1], F. Zaoui[1], Y. Audouin[1], P. Prodanovic[2], J. Fontaine[1], P. Tassi[1], R. Ata[1]
[1] EDF R&D – National Laboratory for Hydraulics and Environment (LNHE), 6 quai Watier, 78401 Chatou, France
[2] Riggs Engineering Ltd., 1240 Commissioners Road West, London, Ontario, N6K 1C7, Canada
cedric.goeury@edf.fr

Abstract— **Derivative-free optimization methods are typically considered for the minimization/maximization of functions for which the corresponding derivatives neither are available for use nor can be directly approximated by numerical techniques. Problem of this type are common in engineering optimization where the value of the cost function is often computed by simulation and may be subject to statistical noise or other form of inaccuracy. In fact, expensive function evaluations would prevent approximation of derivatives, and, even when computed, noise would make such approximations less reliable. Thus, the objective of this work is to implement an efficient heuristic-designed procedure in order to find optimal solution when using TELEMAC-2D to assess a hydrodynamic performance. Two examples are given dealing with a shape optimization and a model calibration. In both cases the underlying optimization problems are solved by coupling a population-based metaheuristic to the numerical model with the help of TelApy [1]. For the shape optimization problem, some design variables define the geometrical configuration of the structure whose optimal configuration is not known a priori. The shape optimization problem consists of finding an optimum position of the slots of a typical fish passage. The second application case focuses on calibration problem. In fact, calibrating a hydrodynamic model (here the Gironde estuary site) is typically an engaged and difficult process due to the complexity of the flows and their interaction. In this paper, both friction and tidal are highlighted. Theoretically, Particle Swarm Optimization algorithm does not ensure to find optimal solutions but in practice it performs very well. Moreover it does not rely on gradient computations as it does not assume the problem to be differentiable. Finally its convergence is fast enough in comparison with other algorithms even when coupled with TELEMAC-2D.**

I. Introduction

Optimization is an area of critical importance in engineering and applied sciences. When designing products, materials, factories, production processes, manufacturing or service systems, and financial products, engineers strive for the best possible solutions, the most economical use of limited resources, and the greatest efficiency. Although the problem of minimizing or maximizing a differentiable function is a frequently met problem, the golden age of optimization has been enabled by developments in the three main areas: computing capability, data, and methods. The optimization former can be very different according to the form of the cost function to be minimised (convex, quadratic, nonlinear, etc.), its regularity and the dimension of the space studied. Derivative-free optimization (DFO) methods are typically considered for the minimization/maximization of functions for which the corresponding derivatives neither are available for use nor can be directly approximated by numerical techniques. Problem of this type are common in engineering optimization where the value of the cost function is often computed by simulation and may be subject to statistical noise or other form of inaccuracy. In fact, expensive function evaluations would prevent approximation of derivatives, and, even when computed, noise would make such approximations less reliable. Thus, the objective of this work is to implement an efficient heuristic-designed procedure in order to find optimal solution when using TELEMAC-2D to assess a hydrodynamic performance.

Section II and III introduce the principle of the Derivative-free optimization algorithm and the software tools used for this work respectively. Section IV is dedicated to model results obtained from several different hydraulic applications: calibration and shape optimization. Finally, Section V, offers some conclusions and outlook.

II. Context and principle

A. Context

Parameter estimation, a subset of the so-called inverse problem, consists of evaluating the underlying input data of a problem from its solution. The Derivative-free optimization is used here for demonstration purpose on two different hydraulic inverse problems:

- Parameter Calibration. Numerical models are nowadays commonly used in fluvial and maritime hydraulics as forecasting and assessment tools for example. Model results have to be compared against measured data in order to assess their accuracy in operational conditions. Amongst others, this process touches on the calibration, verification and validation. In particular, calibration aims at simulating a series of reference events by adjusting some uncertain physically-based parameters until the comparison is as accurate as possible. Calibration is critical to all projects based on numerical models as it takes a very large proportion of the project lifetime. Thus, in this work, a real estuary configuration is presented and calibrated using measurement data.
- Shape optimization configuration. Applications of shape optimization to hydraulic engineering are rare, especially for the cases where optimization is used to inform actual engineering design. The Derivative-free

optimization is used here to present a methodology that can be applied by end users working on their own shape optimization problems in the fields of river and coastal hydraulics. For example, applications of shape optimization could include determining the optimal layout of a groyne field along a coastline that could address sedimentation and navigation issues, the shape and orientation of a breakwater protecting a harbour or a marina subject to various environmental and economic constraints, as well as many others.

B. *Cost function formulation*

Thereafter, all model parameters constitute the n-components of the control vector $X = (X_i)^T, \forall \mathrm{i} \in [1, \ldots \mathrm{n}]$.

The optimization of hydraulic problems is a parameter estimation or reverse method used to simulate a series of reference events by adjusting uncertain physically-based parameters contained in the control vector X to produce a solution that is as accurate as possible. Therefore, the optimal search for the control vector takes the form of an objective or cost function $J(X)$ given in (1).

$$\mathrm{J(X)} = \frac{1}{2}\big(\mathrm{Y} - \mathrm{H(X)}\big)^{\mathrm{T}}\mathrm{R}^{-1}\big(\mathrm{Y} - \mathrm{H(X)}\big) \qquad (1)$$

where the components of X represents parameters to be designed / calibrated, Y is the target state/observation vector, H is an operator enabling the passage of the parameter space (where the vector X lives) to the target state/observation space (where Y lives) such that $Y = H(X)$ and R is a weighted covariance matrix. This is a formulation of the optimal search of control vector X adopted in this work.

Many deterministic optimisation methods are known as gradient descent methods. However, sometimes derivatives neither are available for use nor can be directly approximated by numerical techniques. Problem of this type are common in engineering optimization where the value of the cost function is often computed by simulation and may be subject to statistical noise or other form of inaccuracy. In fact, expensive function evaluations would prevent approximation of derivatives, and, even when computed, noise would make such approximations less reliable. In such cases, Derivative-free optimization methods are particularly useful.

In this paper, the Particle Swarm Optimizer (PSO) has been used to solve the hydraulic optimization problem.

C. *Particle Swarm Optimizer*

1) *Informal description*

Particle swarm optimization is a population-based stochastic optimization technique developed by [2], inspired by social behaviour of bird flocking or fish schooling. PSO shares many similarities with evolutionary computational techniques such as Genetic Algorithms [3]. The system is initialized with a population of random solutions and searches for optima by updating generations. In PSO, the potential solutions, called particles, fly through the search space by following the current optimum particle. In fact, for each particle, it is possible to evaluate the cost function value given by Eq.1. Then, the global optimum point of the particle swarm, i.e. the one having the smallest cost function value is looked for. This is useful to compute a velocity for each particle. The particle swarm optimization concept consists of, at each iteration of the algorithm, changing the velocity of each particle towards the best solution. A particle is made of:

- a position inside the search space
- the cost function value at this position
- a velocity (in fact a displacement), which is used to compute the next position
- a memory, that contains the best position (called the previous best) found by the particle
- the cost function value of this previous best

2) *Mathematical formulation*

In a search space of dimension d, the swarm particle i has a location and velocity (in fact a displacement) vector $\overrightarrow{x_i} = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$ and $\overrightarrow{v_i} = (v_{i,1}, v_{i,2}, \ldots, v_{i,d})$ respectively. The quality of its position is determined by the value of the objective function at this point. Moreover, the best position by which the particle has already passed, denoted $\overrightarrow{Pbest_i} = (Pbest_{i,1}, Pbest_{i,2}, \ldots, Pbest_{i,d})$, is kept in memory. Another “best” value that is tracked by the particle swarm optimizer is the best value, obtained by any particle of the swarm is denoted as

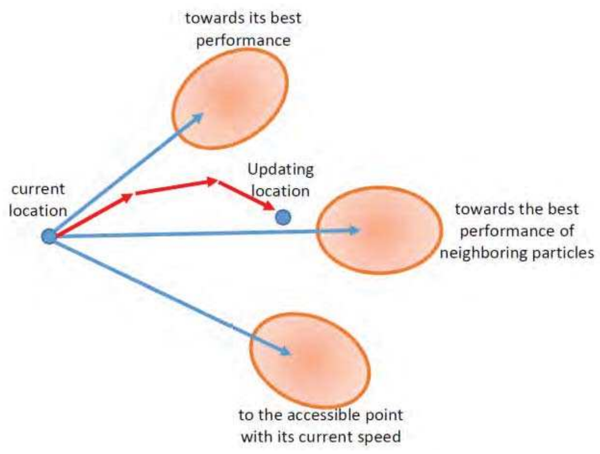
$$\overrightarrow{Gbest} = (Gbest_1, Gbest_2, \ldots, Gbest_d).$$

At the iteration $t+1$, the new particle position $\overrightarrow{x_i^{t+1}}$is computed as expressed in Eq. 2.

$$\begin{cases} x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1} \\ v_{i,j}^{t+1} = \omega v_{i,j}^{t} + c_1 r_1 \big(Pbest_{i,j}^{t} - x_{i,j}^{t}\big) \\ \ldots + c_2 r_2 \big(Gbest_j - x_{i,j}^{t}\big) \end{cases} \qquad (2)$$

where ω is a constant called inertia coefficient, c_1 and c_2 are two constants representing acceleration coefficients, r_1 and r_2 are two numbers randomly generated at each iteration and dimension from the uniform distribution $U[0; 1]$. In this work, the parameters ω, $c_1$and c_2 are set to the default value 0.5.

As presented in Eq.2 and displayed in Figure 1, the particle displacement is governed by the following components: inertia term (ωv_i^j), cognitive and social component respectively $c_1 r_1 \big(Pbest_i^j - x_i^j\big)$ and $c_2 r_2 \big(Gbest^j - x_i^j\big)$.


towards its best performance
Updating location
current location
towards the best performance of neighboring particles
to the accessible point with its current speed
Figure 1. Particle displacement.

Then, the variables $\overrightarrow{Pbest_i}$ and, $\overrightarrow{Gbest}$ are determined using Eq. 3 and 4.

$$\overrightarrow{Gbest} = arg \min_{\overrightarrow{Pbest_i}} J(\overrightarrow{Pbest_i}), \forall\ i \in [1, N] \qquad (3)$$

$$\overrightarrow{Pbest^{t+1}} = \begin{cases} \overrightarrow{Pbest_i}, \text{if } J\left(\overrightarrow{x_i^{t+1}}\right) \geq \overrightarrow{Pbest_i} \\ \overrightarrow{x_i^{t+1}} \text{ else} \end{cases} \qquad (4)$$

where J denote the cost function defined in Eq. 1.

This process is summarized in the algorithm presented in Table 1.

```
1-Initialisation of the swarm composed by $\boldsymbol{N}$ particles: pick a random position and velocity
2-Compute the particle positions
3-For each particle $\boldsymbol{i}$, $\overrightarrow{\boldsymbol{Pbest_i}} = \overrightarrow{\boldsymbol{x_i}}$
4-Compute $\overrightarrow{\boldsymbol{Gbest}}$
5-While the stop criterion is not satisfied do:
6-   Compute the particle displacement (Eq. 2)
7-   Evaluate the particle positions (call of hydraulic solver)
8-   update , $\overrightarrow{\boldsymbol{Pbest_i}}$ and , $\overrightarrow{\boldsymbol{Gbest}}$ (Eq. 3 and Eq. 4)
9-End
```

Table 1. Particle Swarm Optimization Algorithm

III. SOFTWARE TOOLS

The particle swarm optimization algorithm presented in the previous section (section II) combines different fields such as optimisation, numerical analysis, parameter estimation, and free surface flow hydraulics. The software implementation of the algorithm has to be designed for different architectures with reusable components. This study is performed by coupling the hydrodynamic solver TELEMAC-2D and the toolkit library PYSWARM for particle swarm optimization in python within the SALOME platform, through the component TelApy of the TELEMAC system.

A. The SALOME platform

SALOME is an open source platform (www.salome-platform.org) for pre and post processing of numerical simulations, enabling the chaining or the coupling of various software tools and codes. SALOME is developed by EDF, the CEA and OPENCASCADE S.A.S. under the GNU LGPL license. It is based on an open and flexible architecture with reusable components, which can be used together to build a computation scheme assembling each module or external codes together through specific communication protocols. In our case, the TELEMAC-2D model is driven through the TelApy component and dynamically linked to PYSWARM library within SALOME (See Fig. 2). In fact, all the components within SALOME can be used together with the YACS module which builds a computation scheme and call each module and makes them communicate. In our case TELEMAC-2D and PYSWARM are working together within this platform. Moreover, the platform supports advanced generation of numerical model geometry through it extensive CAD modelling computational engine, thus making it applicable to a wide variety of studies and applications. Its meshing capabilities are also extensive, allowing a user to generate meshes using common 2D and 3D formats. Moreover, the SALOME platform has provided its users access to all of its functionalities through an integrated Python interface. These features make for the SALOME platform an ideal choice for a tool set used shape optimization studies where meshes need to be generated automatically for each new designed parameters.

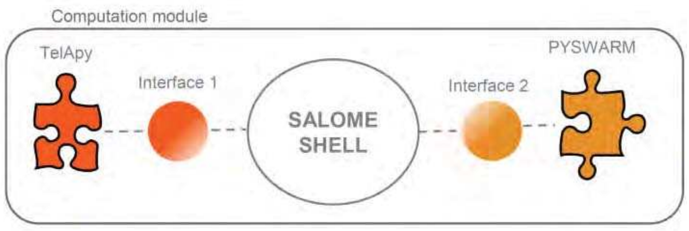


Figure 2. The SALOME composition linking TelApy to PYSWARM library

B. The TelApy component of the TELEMAC system

The recently implemented TelApy component is distributed with the open source TELEMAC system (www.opentelemac.org). It aims at providing python source code that wraps and controls a TELEMAC simulation through a Fortran API (Application Program Interface) [1]. The API's main goal is to have control over a simulation while running a case. For example, it allows the user to hold the simulation at any time step, retrieve some variables and / or change them. The links between the various interoperable scientific libraries available within the Python language allows the creation of an ever more efficient computing chain able to more finely respond to various complex problems. The TelApy component has the capability to be expended to new types of TELEMAC simulations including high performance computing for the computation of uncertainties, other optimization methods, coupling, etc.

C. The particle swarm optimization PYSWARM

The population-based metaheuristic algorithm used for this work is a Particle Swarm Optimizer written in Python that is a fork of the open source module PYSWARM[1]. PSO defines and iteratively improves a set of candidates (particles) towards optimality. Theoretically, PSO does not ensure to find optimal solutions but in practice it performs very well. Moreover it does not rely on gradient computations as it does not assume the problem to be differentiable.

IV. Applications

A. Shape Optimization application

1) Numerical configuration

The shape optimization problem used in this work consists of finding an optimum position of the slots of a typical fish passage. The geometry of the fish passage used in this example is obtained from [4], where a similar problem is solved (albeit in a different way). The fish passage consists of ten identical compartments, with each having two slots (See Fig. 3). The slots are referred to in this work as upper and lower. The position of the slots for any given shape is specified with four variables representing the center point of the upper (x_U, y_U) and lower (x_L, y_L) slot, respectively (See Fig. 3). A combination of the four values thus defines a particular shape of the flume. Thus, the objective is to design the fish passage shape by changing the position of the slots (x_U, y_U, x_L, y_L) in order to obtain a desired target velocity in the channel. The optimization problem is to select the position and length of the upper and lower slots, such that it optimizes the objective function while satisfying the problem constraints. In fact, in the numerical optimization the search space is restricted to a rectangular zone bounded by (x_a, y_a) on the lower left, and (x_b, y_b) on the upper right. A constraint is specified in the optimization to ensure the slots are spaced at least $\Delta x = x_L - x_U$ apart horizontally, and at least $\Delta y = y_L - y_U$ vertically. This constraint is required in order to prevent the numeric optimizer from selecting invalid and physically irrelevant geometries (such as one where the upper and lower slots touch or overlap).

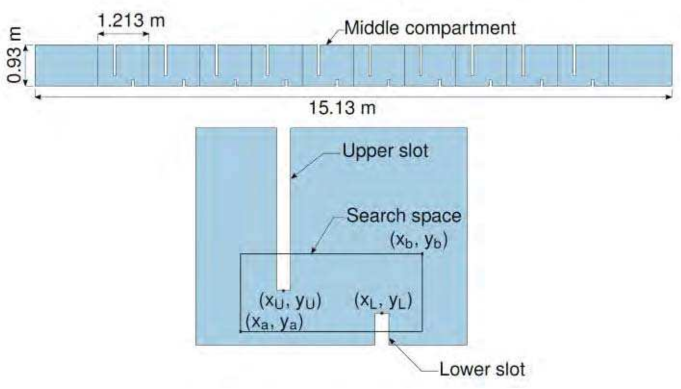


Figure 3. Geometry definition of the fish flume test case

The particulars of the problem simulated here are (units in meters, unless otherwise specified): Search space $x_a = 6.653$, $x_b = 7.260$, $y_a = 0.050$, and $y_b = 0.243$, width of the slots $w = 0.063$; channel slope $S_0 = 0.05$; bottom Chezy friction coefficient $f = 57.36\ m^{1/2}/s$; initial water depth $h_0 = 0.50$; discharge in the flume $Q_0 = 0.065\ m^3/s$; constraint values $\Delta x = 0.1$, and $\Delta y = 0.05$.

2) Optimization problem definition

One way to validate the tool chain developed in this work is to artificially set up a desired state. The aim of this optimization configuration is to find optimal shape design based on a numerically generated synthetic data from the so-called "identical-twin-experiment", in which true state is known. The initial (starting shape) used in the optimization process is defined as, $(x_U, y_U, x_L, y_L)_i = (6.7, 0.23, 7.23, 0.15)$, while the target shape to recover is defined as $(x_U, y_U, x_L, y_L)_t = (6.927, 0.147, 7.168, 0.054)$, where the subscript i, tdenotes the initial and target shape respectively. The optimization process is then started with some initial state, with an end goal to recover the specified (or desired) shape. In order to solve the shape optimization problem applied in hydraulic engineering the end user is required to possess a set of tools that can:

- i) simulate a given shape and obtain a simulated system state,
- ii) extract results from the simulated system state and obtain a value of a pre-defined objective function,
- iii) try a new shape, and simulate its system state,
- iv) carry out as many new iterations until a global optimum is found.

3) Cost function formulation

In the cost function formulation given in Eq. 1, the target state vector is the x and y components of the flow velocity, in the middle compartment (See Fig. 3), extracted from the steady state solution of the target shape such as $Y = \begin{bmatrix} u_t \\ v_t \end{bmatrix} = H(x_U, y_U, x_L, y_L)$.

The operator H enabling the passage of the parameter space (slots coordinates) to the target state space (velocity fields) consists of a tool chain that can automatically:

- generate new shapes (which requires create a new mesh, assigning bathymetry/topography to the mesh, assigning initial and boundary conditions) that are ready to be used in a numerical model,
- call the hydraulic solver TELEMAC-2D,
- and extract from the steady state solution of the shape the x and y components of the flow velocity (u, v) in the middle compartment.

Thus, the main issue for the shape optimization is to have a tool allowing to mesh automatically new shapes. The SALOME platform supports advanced generation of numerical model geometry through its extensive CAD modelling computational engine, thus making it applicable to a wide variety of studies and applications. The SALOME platform has provided its users access to all of its functionalities through an integrated Python interface. These

[1] https://github.com/fzao/pyswarm

features make the SALOME platform an ideal choice for a tool set used in the shape optimization studies where meshes need to be generated automatically for each new designed parameters.

And finally, R is diagonal matrix containing node weighting according to their area.

4) *Numerical results*

In this work, the optimizer selects and tries new shapes during its course of execution. Each iteration of the optimizer requires a TELEMAC-2D solution of a shape where positions of the slots are evaluated for each considered particle. In this example, the swarm of PSO optimizer is composed of 10 particles and the maximum number of iterations is set to 20. After specifying initial and target shapes the optimization simulations are carried out. Table 2 shows the results of the simulations.

Simulation shape	x_U [m]	y_U [m]	x_L [m]	y_L [m]	$J(x)$ [-]
Initial	6.700	0.230	7.230	0.150	0.137
Target	6.927	0.147	7.168	0.054	0.0
optima	6.924	0.152	7.193	0.052	0.0033

Table 2. Slot positions and cost function at initial, target and optima configurations

Figure 4 presents the generated optimum graphically and optimizer convergence plots is shown in Figure 5.

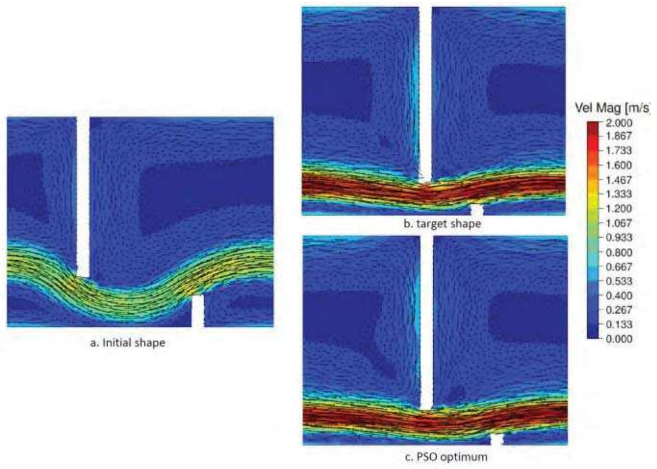


Figure 4. Fish flume at initial (a), target (b) and optima (c) configurations

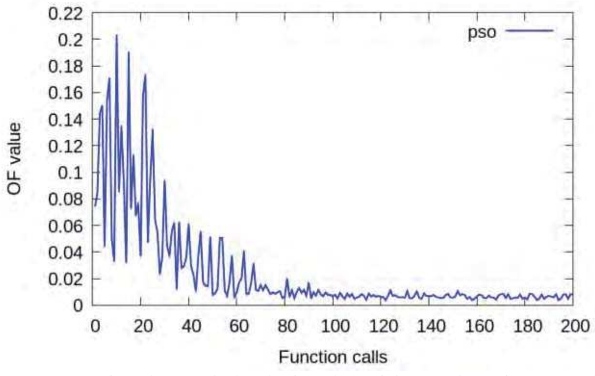


Figure 5. Cost function evolution in function of the number of computation calls

As shown in Figure 4, the fish flume optimum structure is much closer to the target shape than the initial one. As expected, the final results emphasises the efficiency of the parametric shape optimization tool which is able to deliver optimal structure design for hydraulic engineering applications. However, some slot position differences can be observed (see Tab. 2). The defined objective function does not seem to be sensitive to small fluctuation in the position of the slots. Given the shape optimization tool kit is working as intended, a relevant question to pose is what are the effects of the four design variables on the previously defined optimization problem. This question is answered by carrying out a global sensitivity analysis using Morris's method [5] on the four design variables (x_U, y_U, x_L, y_L), and to analyze how they influence the defined objective function $J(x)$ in the fish flume case.

5) *Sensitivity Analysis*

The sensivity analysis aims at quantifying the relative importance of each input parameter of a model. The variance-based methods aim at decomposing the variance of the output to quantify the participation of each variable when these ones are considered as independents. Generally, these techniques compute sensitivity indices called Sobol Indices [6]. The definition of Sobol indices is a result of the ANOVA (ANalysis Of VAriance) variance decomposition. Morris's method, unlike Sobol's method, only provides qualitative answers regarding parameter interactions but it does so with much less model evaluations. In fact, Morris's method measures global sensitivity using a set of local derivatives (elementary effects) taken at discrete points sampled through the parameter space. Each parameter is perturbed along a pre-defined grid to create a trajectory through the parameter space. For a given model with d parameters, one trajectory will contain d perturbations. Each trajectory yields an estimate of the elementary effect for each parameter (ratio of the change in model output to the change in parameter). Once trajectories are sampled, the resulting set of elementary effects are then averaged to give an estimate of total order effects [5]. The standard deviation of the elementary effects describes the variability through the parameter space, and thus describes the degree to which interactions are present. Total order effects are described with a parameter μ; the higher the value for a particular parameter, the more influential that parameter is. The degree of interaction is captured with a parameter σ; the higher the value for a particular parameter, the more the parameter in question interacts with other parameters.

Thus, in this work, the variables (or parameters) considered in the Morris's method are the four design variables of the shape optimization problem (x_U, y_U, x_L, y_L). The bounds of each design variable had to be individually specified such that $(x_U, y_U) \epsilon [6.653,6.994] \times [0.129,0.242]$ and $(x_L, y_L) \epsilon [7.097,7.250] \times [0.010,0.079]$, to make sure the set of samples generated by Morris's method actually meets the constraints of the optimization problem. For the sampling of the parameter space using Morris's method, the number of trajectories, r, is set at 10. The number of grid points, p, to sample the parameter space in each dimension is also set at 10. From the theory embedded in Morris's method, the number of parameter samples that is generated is $r \times (d+1) = 10 \times (4+1) = 50$. Application of Morris's method, in combination with the tool chain built in this work, produce the Morris's effects plot shown in Figure 6.

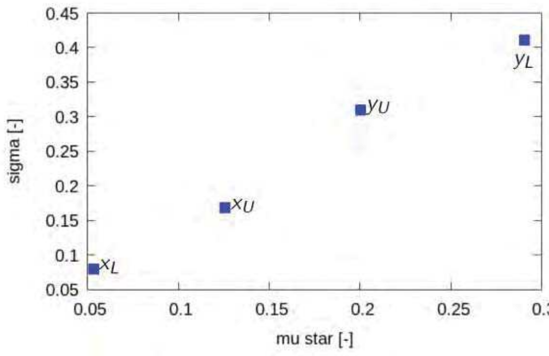


Figure 6. Global sensitivity analysis of design variables using Morris's method

From Morris's effects plot shown in Figure 6, it is readily discernible that the y coordinates of the fish slots are the most influential variables, together with the fact that they are also the ones that are most likely to interact with each other. This conclusion is consistent with the findings discovered through the numerical results presented in Figure 4, where it was found that varying the x coordinates of the slots have less influence on the velocity magnitude.

B. Calibration application

1) Context

The Bordeaux harbour, located in the largest estuary in Western Europe, faces many challenges of development in the Gironde estuary area. It must simultaneously manage the estuary natural constraints and improve its capacity of reception of larger ships in the next two years to answer the growing international market demand. In fact, the increasing use of maritime transport leads to an increase in ship size in order to minimize the transport costs in terms of budget and time. On the other hand the dimension of access channels and harbours cannot follow the expansion rate of the vessels. Thus, in order to satisfy the demand of the market for increasing ships size, while ensuring navigation safety, the water-depth evolution in the estuary needs to be predicted with a maximal accuracy. Numerical models are nowadays commonly used in fluvial and maritime hydraulics as forecasting and assessment tools for example. Model results have to be compared against measured data in order to assess their accuracy in operational conditions. Amongst others, this process touches on the calibration, verification and validation. In particular, calibration aims at simulating a series of reference events by adjusting some uncertain physically based parameters until the comparison is as accurate as possible. Calibration is critical to all projects based on numerical models as it requires a very large proportion of the project lifetime. The objective of this work is to implement an efficient calibration algorithm, capable of processing measurements optimally, to estimate the partially known or missing parameters (bathymetry, bed friction, inflow discharge, tidal parameter, initial state, etc.).

2) Numerical configuration and available data

The Gironde is a navigable estuary in southwest France and is formed from the meeting of the rivers Dordogne and Garonne just downstream of the centre of Bordeaux, it is the largest estuary in western Europe. The hydraulic model used in this work covers approximately $195\ km$ between the fluvial upstream and the maritime downstream boundaries conditions representing an area of around $635\ km^2$. The finite element mesh is composed of 173781 nodes (see Fig. 7). The mesh size varies from $40\ m$ within the area of interest, the navigation channel, to $750\ m$ offshore (western and northern sectors of the model). As shown in Figure 7, six friction areas are considered in the hydraulic model.

The boundary conditions along the marine border of the model have been set up using depth-averaged velocities and water levels from the Legos numerical model TUGO dataset (46 harmonic constants). Surge data, describing the difference between the tidal signal and the observed water level, are taken into account using a data file that comes from Hycom2D model of the SHOM [7]. Surface wind data is also considered in the model to simulate the flow under wind blowing conditions. A flow discharge is imposed upstream of the Gironde estuary model on the Garonne and the Dordogne rivers. Time series are available for these two liquid boundaries.

Several observation stations are available in the region of interest. These stations measure the free surface flow evolution every 60 seconds at the The Verdon, Richard, Lamena, Pauillac, Fort Médoc, Ambes, The Marquis, Bassens and Bordeaux locations (see Fig. 7). For this study, observation results are used over a 36 hours period from August 12th to 14th, 2015.

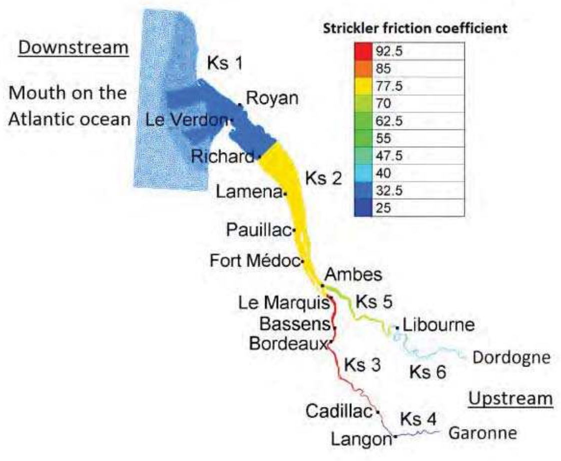


Figure 7. Model mesh with friction coefficient areas and observation station locations

3) Sensitivity analysis

Calibrating a hydrodynamic model (here for a real tidal site) is typically an engaged and difficult process due to the complexity of the flows and their interaction with the shoreline, the bathymetry, islands, etc. Thus, it is essential to understand in depth the relationship between the modelling calibration parameters and the simulated state variables which are compared to the observations. In this case, the identification of the most influential input parameters by sensitivity analysis has been led to target the calibration parameters when observations are available. In particular, both friction and tidal amplification were highlighted.

a) friction coefficient

Friction comes into the momentum equations of the shallow water equations and is treated in a semi-implicit form within TELEMAC-2D [8]. The two components of friction force are given in Eq. (6).

$$\begin{cases} F_x = -\frac{u}{2h} C_f \sqrt{u^2 + v^2} \\ F_y = -\frac{v}{2h} C_f \sqrt{u^2 + v^2} \end{cases} \tag{3}$$

where h is the water depth, C_f a dimensionless friction coefficient and u and v are the horizontal x and y components of the current velocity.

The roughness coefficient often takes into account the friction by the walls on the fluid or other phenomena such as turbulence. Thus it is difficult to define directly from available data and must be adjusted using the water surface profiles measured for a given flow rate.

b) Tidal amplification parameter

Tidal characteristics are imposed using a database of harmonic constituents to force the open boundary conditions. For each harmonic constituent, the water depth h and horizontal components of velocity u and v are calculated, at point M and time t by Eq. (7).

$$\begin{cases} F(M,t) = \sum_i F_i(M,t) \\ F_i(M,t) = \\ \qquad f_i(t) A_{F_i}(M) \cos\left(\frac{2\pi t}{T_i} - \varphi_{F_i}(M) + u_i^0 + v_i(t)\right) \end{cases} \tag{7}$$

where F is either the water level (referenced to mean sea level) z_S or one of the horizontal components of velocity u or v, i refers to the considered constituent, T_i is the period of the constituent, A_{F_i} is the amplitude of the water level or one of the horizontal components of velocity, $\phi_{F_i}$is the phase, $f_i(t)$ and $v_i(t)$ are the nodal factors and u_i^0 is the phase at the original time of the simulation. The water level and velocities of each constituent are then summed to obtain the water depths and velocities for the open boundary conditions (8).

$$\begin{cases} h = \alpha \sum z_{Si} - z_f + z_{mean} + \gamma \\ u = \beta \sum u_i \\ v = \beta \sum v_i \end{cases} \tag{4}$$

where z_f is the bottom elevation and $z_{mean}$the mean reference level. In Eq. (8), the tidal amplitudes multiplier coefficient of tidal range and velocity, respectively α and β, at boundary locations and the sea level correction γ are assumed to be the tidal calibration parameters [9].

c) Analysis of variance

The sensivity analysis has been carried out based on the computation of Sobol sensitivity indices according to the methodology presented in [11]. In this paper, we investigate the effect of three sources of uncertainty, the friction coefficients, the tidal amplification coefficients along the marine boundaries (α and β) and the mean water level correction coefficient γ. The source quantification of the uncertain variables, arbitrarily chosen, is summarized in Table 3.

Parameter	Probability density function
K_1 [$m^{1/3}s^{-1}$]	$U[30.4; 45.6]$
K_2 [$m^{1/3}s^{-1}$]	$U[64; 96]$
K_3 [$m^{1/3}s^{-1}$]	$U[80; 100]$
K_4 [$m^{1/3}s^{-1}$]	$U[20; 30]$
K_5 [$m^{1/3}s^{-1}$]	$U[64; 96]$
K_6 [$m^{1/3}s^{-1}$]	$U[36; 54]$
α and β [-]	$N(1; 0.07) \in [0.8; 1.2]$
γ [m]	$N(0.4438; 0.0295) \in [0.355; 0.532]$

Table 3. Source Quantification of uncertain variables

To handle the sensitivity analysis, it is important to run a lot of simulations in order to have reliable results. In this work, around 15,000 Monte-Carlo computations have been carried out based on TELEMAC-2D through the SALOME platform described in [10].

Figure 8 displays the total Sobol sensitivity indices obtained at the Bordeaux observation station.

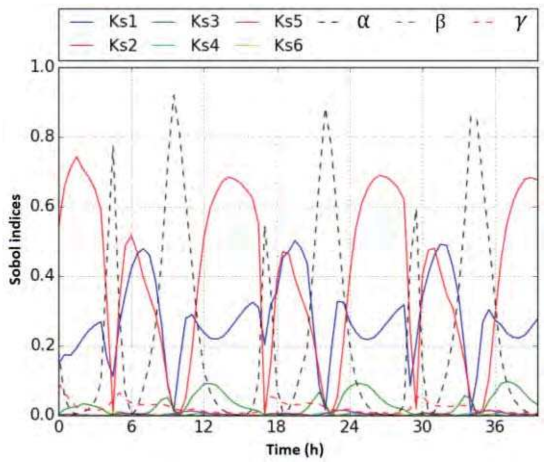


Figure 8. Total Sobol indices at the Bordeaux station

As shown by the sensitivity analysis, the most influent variables on the water depth variation are the friction coefficients (K_1, K_2, K_3) (in $m^{1/3}s^{-1}$), the tidal amplitudes multiplier coefficient of tidal range α and the sea level correction γ (in m). The other variables can be considered negligible in comparison. These results depend, of course, on the hypothesis on the input random variables and especially on the choice of their distributions. Consequently, the calibration of the model is focused on these parameters.

4) Parameter calibration

In the cost function formulation given in Eq. 1, the observation vector is the free surface flow evolution extracted every 60 seconds at the The Verdon, Lamena, Pauillac, Fort Médoc, Bassens and Bordeaux locations. The observation operator H represents a call to the hydraulic solver and the observation covariance matrix R contains small value terms leading to represent a huge confidence on the observation value. At each iteration of the optimization algorithm, 448 particles fly through the search domain. The algorithm is stopped after 10 iterations. Moreover, the PSO algorithm results are compared with the results obtain from the gradient based method described in [11]. The obtained results are summarized in the Table 4.

parameters	K_1	K_2	K_3	α	γ	$J(X)$
Initial	19	50	50	0.75	-0.0562	6,34e6
PSO optimum	40.8	58.3	107	1.00	0.502	1,63e5
Gradient optimum	39.2	60.5	98.5	0.99	0.497	1,64e5

Table 4. parameter values and cost function at initial, gradient based and derivative free optima configurations

Calibrated parameters (see Tab. 4) with derivative free and gradient based approaches are not so far. The number of function calls is 224 for the gradient based method and 4480 for the PSO algorithm. In fact, the derivative information in optimization process allows to drastically reduce the number computations.

Figure 9 displays the results of the automatic calibration over a 36 hours period.

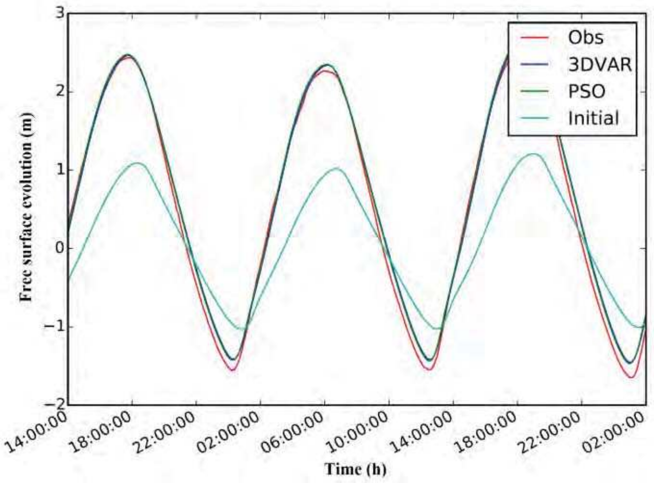


Figure 9. Free surface evolution at the Lamena station

As shown in Figure 9, the water surface profiles calculated are much closer to the measurements than the initial model calibration. The final results emphasises the efficiency of the automatic calibration tool in the framework of a maritime configuration. Moreover, the computation time is a crucial point from operational point of view. Thus, the algorithmic optimization tool implemented in this work has been written to make use of multiprocessor parallelism in order to be efficient and compatible with industrial needs. In fact, at each PSO iteration, the computation of each particle is an independent hydraulic state. Thus, the particle swarm can be evaluated in parallel. As TELEMAC is also a MPI-based parallel code, different configurations for parallelism are possible. In this work, the particle swarm is distributed on cluster to run in parallel and the particle computation is sequential.

V. CONCLUSIONS

Derivative-free optimization methods are typically considered for the minimization/maximization of functions for which the corresponding derivatives neither are available for use nor can be directly approximated by numerical techniques. Problem of this type are common in engineering optimization where the value of the cost function is often computed by simulation and may be subject to statistical noise or other form of inaccuracy. In fact, expensive function evaluations would prevent approximation of derivatives, and, even when computed, noise would make such approximations less reliable. Thus, the objective of this work is to implement an efficient heuristic designed procedure in order to find optimal solution when using TELEMAC-2D to assess a hydrodynamic performance. Two examples are given dealing with a model calibration and a shape optimization. In both cases the underlying optimization problems are solved by coupling a population-based metaheuristic to the numerical model with the help of TelApy [1]. The population-based metaheuristic is a Particle Swarm Optimizer written in Python that is a fork of the open source module PYSWARM. PSO defines and iteratively improves a set of candidates (particles) towards optimality. Theoretically, PSO does not ensure to find optimal solutions but in practice it performs very well. Moreover it does not rely on gradient computations as it does not assume the problem to be differentiable. Finally its convergence is fast enough in comparison with other algorithms even when coupled with TELEMAC-2D. Future works will include the multicriteria optimization and combination of DFO and gradient based algorithms to take advantage of both approaches.

REFERENCES

[1] C. Gœury, Y. Audouin, and F. Zaoui, “User documentation v7p3 of TelApy module,” 2017

[2] R. C. Eberhart, J. Kennedy. “A new optimizer using particle swarm theory,” 6 th International Symposium on Micromachine and Human Science, Nagoya, Japan, pp. 39-43, 1995

[3] W.F. Abd-El-Wahed, A.A. Mousa, M.A. El-Shorbagy, “Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems”, Journal of Computational and Applied Mathematics, 235(5), 1446-1453, 2011.

[4] L. Alvarez-Vasquez, A. Martinez, M. Valquez-Mendez, M. Villar, “Numerical resolution of a shape optimization problem in hydraulic engineering”, European Conference on Computational Fluid Dynamics, P. Wesseling and J. Periaux, eds., TU-Delft, pp. 1-13.

[5] J. Herman, J. Kollat, P. Reed, T. Wegener, “Technical Note: Method of Morris effectively reduces the computation demands of global sensitivity analysis for distributed watershed models.” Hydrology and earth system sciences, 17, pp.2893–2903.

[6] I.M Sobol′, “Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates”, Mathematics and Computers in Simulation, 55(1-3), 271-280, 2001.

[7] E. P. Chassignet, H. E. Hurlburt, O. M. Smedstad, G. R. Halliwell, P. J. Hogan, A. J. Wallcraft, R. Baraille, and R. Bleck, “The hycom (hybrid coordinate ocean model) data assimilative system”, Journal of Marine Systems, 65(1) :60-83, 2007.

[8] J-M. Hervouet, “Hydrodynamics of Free Surface Flows”, Wiley, 2007, pp. 83–130.

[9] C. T. Pham, F. Lyard, “Use of tidal harmonic constants databases to force open boundary conditions in TELEMAC”, Proceedings of the 19th Telemac-Mascaret User Club, 2012.

[10] C. Goeury, T. David, R. Ata, S. Boyaval, Y. Audouin, N. Goutal, A.-L. Popelin, M. Couplet, M. Baudin, R. Barate, “Uncertainty Quantification on a real case with TELEMAC-2D”, Proceedings of the 22nd Telemac-Mascaret User Club, 2015.

[11] C. Gœury, A. Ponçot, J.-P. Argaud, F. Zaoui, R. Ata, and Y. Audouin, “Optimal calibration of TELEMAC-2D models based on a data assimilation algorithm”, 24th Telemac-Mascaret User Conference, Graz, Austria, pp. 73- 80, October 2017