V European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2010
J. C. F. Pereira and A. Sequeira (Eds)
Lisbon, Portugal,14-17 June 2010

UNCERTAINTY QUANTIFICATION BASED ON FORWARD SENSITIVITY ANALYSIS IN SISYPHE (ECCOMAS CFD 2010)

Jan Riehme*, Rebekka Kopmann†, and Uwe Naumann*

*Dept. of Computer Science, University of Hertfordshire
College Lane, Hatfield AL10 9AB, UK
and
LuFG Informatik 12: Software and Tools for Computational Engineering,
RWTH Aachen University, 52056 Aachen, Germany
e-mail: {riehme, naumann}@stce.rwth-aachen.de

†Bundesanstalt für Wasserbau
Kußmaulstr. 17, 76176 Karlsruhe, Germany
e-mail: rebekka.kopmann@baw.de

Key words: algorithmic/automatic differentiation, reliability analysis, morphodynamic simulation

Abstract. *We present a tangent-linear version of the two-dimensional morphodynamic model Sisyphe[1] generated by algorithmic (also known as automatic) differentiation. Our group has been developing a differentiation-enabled research prototype[16] of the NAG[2] Fortran compiler for several years. The resulting differentiated variant of Sisyphe is used to calculate the sensitivities of the evolution with respect to different input parameters, which are assumed to be uncertain. These sensitivities allow to perform a first-order reliability analysis that estimates the uncertainties regarding chosen input parameters[10].*

In a laboratory experiment, the development from a initial flat bed to a sloped cross section in a 180 degree bend is predicted with respect to four different input parameters. The results of the reliability analysis allow a ranking of the uncertain input parameters concerning their level of uncertainty. Moreover, instead of calculating just a single value for the evolution at one point in time and space, a probability distribution of the evolution is derived from the morphodynamic model. The method is shown to be applicable to real-world cases demonstrated for a 10 km long stretch of the river Danube.

We have observed a good qualitative match with results from Monte-Carlo simulations. The analysis of quantitative discrepancies are the subject of ongoing work. For the laboratory experiment the Monte-Carlo simulation took roughly 5 hours of computation time, since 100 model evaluations were required. In contrast, for the first-order reliability analysis method one evaluation of the sensitivities of the model is sufficient, taking only 22 minutes.

[1] developed at E.D.F. Laboratoire National D'Hydraulique et Département Environnement; `www.edf.com`

[2] Numerical Algorithms Group Ltd.; `www.nag.com`

1 INTRODUCTION

Uncertainties are unavoidable in numerical modelling due to the deficient description of the physical processes and the imprecision of model parameters. In morphodynamic modelling input parameters are uncertain due to measurement errors, natural variability, or unsatisfactory parameterisation. However, the propagation of uncertainties in the input data might have serious influence on the simulation results. Therefore, it is necessary to quantify their contributions to the model results in order to appraise their reliability.

Probability distributions of output variables with respect to the uncertainty of input parameters can be computed by a special variant of the widely used Monte Carlo method. Naturally Monte Carlo methods require an enormous number of model evaluations, where a single evaluation can take days or weeks even in modern parallel computing environments.

The *First-Order Reliability Analysis Method* approximates the desired probability distributions by a single evaluation of the sensitivities of output variables with respect to the uncertain input parameters. However, the advantage of a lower computational effort comes at the cost of a significant effort required to derive a version of the model that can compute sensitivities. Hand coding the sensitivity model is time consuming and error prone, especially for larger codes such as Sisyphe [18]. Moreover, keeping the sensitivity model in sync with the ongoing development of the model code itself is usually a major effort.

Compilers for *algorithmic (or automatic) differentiation* augment the original model with sensitivity computing code semi-automatically with often minimal intervention by the user. New versions of the model are differentiated by reapplying the compiler.

The outline of the paper is as follows: Section 2 gives a short introduction to reliability analysis and the *Monte Carlo Confidence Limit method* used for comparison, followed by the First-Order Reliability Analysis Method that requires the differentiation of the model. Section 3 introduces the concept of algorithmic differentiation and delineates the differentiation of the model by the differentiation-enabled NAG Fortran compiler [16]. Applications, observations, and results are given in Section 4 followed by conclusions.

2 RELIABILITY ANALYSIS

Morphodynamic modelling is often used to forecast long-term and large-scale phenomena like river bed erosion of a river stretch. It is not uncommon for the evaluation of such more-dimensional models to take days or even more computing time. As the input data contain uncertainties due to measurement errors or natural variability rating the trustworthiness of those expensive results of more-dimensional model evaluations is coming more and more in focus. Hence information about the effect of the various uncertainties in the input parameters on the model results is required. Reliability analysis can be used to quantify the influence of uncertain input parameters on the state variables.

In this paper the TELEMAC modelling system is used for calculating the two-dimensio-

nal free-surface flow (Telemac-2D [8]) and the sediment transport (Sisyphe [18]). Both modules are coupled internally to capture the interaction between hydrodynamics and morphodynamics. For long-term modelling it could be useful to run Sisyphe stand alone in order to speed up the model evaluations. In that case the hydrodynamics will be read from disc or it is assumed to be in steady state. In this case the influence of the morphodynamics on the hydrodynamics is not taken into account properly. The approach is built on the assumption that for small changes of the river bottom the missing interaction will not falsify the results. For comparison the confidence limits of the bottom evolution calculated by Sisyphe are computed by the First-Order Reliability Analysis Method and the Monte Carlo Confidence Limit method. Both methods are briefly described in sections 2.1 and 2.2.

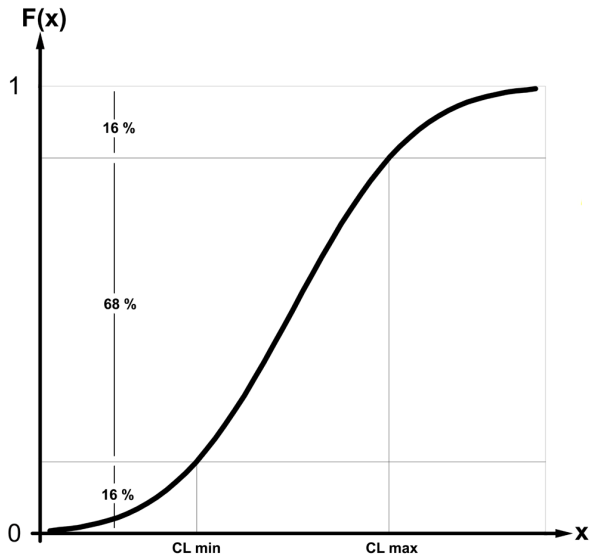


Figure 1: CDF and confidence limits $CL_{\min}$ and $CL_{\max}$ for confidence level $\alpha = 68\%$.

2.1 Monte Carlo Confidence Limit Method

In the Monte Carlo Confidence Limit (MCCL) method [17] the confidence limits are determined approximatively. For a given confidence level α they are obtained from a usually unknown cumulative distribution function (CDF). Figure 1 shows an exemplary CDF and its confidence limits for a confidence level of $\alpha = 68\%$. In order to get the values of the confidence limits for the confidence level α the CDF has to be inverted in the two points $(1 \pm \alpha)/2$. An empirical distribution function (EDF) that approximates the unknown CDF is computed from the results of N model evaluations. Confidence limits are computed by inverting the EDF. The EDF is a staircase function with step length of $1/N$ as shown in Figure 2.

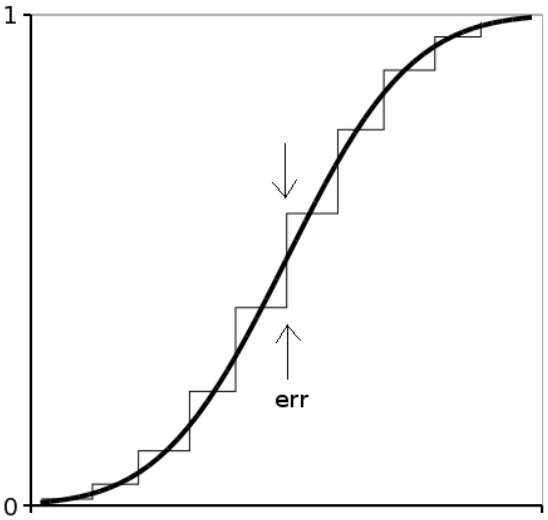


Figure 2: EDF (staircase) as an approximation of the CDF.

If the number N of model evaluations performed is big enough, then the EDF converges to the CDF (law of large numbers). In case of infinitely many simulation runs the approximation error of the EDF

$$err_{EDF} = \frac{1}{N} \tag{1}$$

tends to zero. Additionally, there is a statistical error of the CDF itself, which can be derived from the central limit theorem as

$$err_{CDF} = \sqrt{\frac{1-\alpha^2}{4N}} \quad . \tag{2}$$

This error dominates the approximation error err_{EDF} for large values of N

$$N \gg \frac{2}{(1-\alpha)} \tag{3}$$

and is normally (Gaussian) distributed.

The number N of simulations required by the MCCL method is independent of the number of uncertain parameters. It depends on the chosen confidence level α only. So the value of N should be chosen as a tradeoff between maximising the confidence level α and the available computing time. For the practically most relevant confidence level $\alpha = 95\%$ at least $N = 100$ simulation runs must be performed [17].

A design of experiment generator [17] is used to generate N different input parameter sets forming a probability distribution in the space of the input parameters. The N simulations transform this probability distribution into the space of the output values. Each output value (for example, the evolution in time and space) can be seen as a single random number while its CDF determines the confidence limits.

2.2 First-Order Reliability Analysis Method

The First-Order Reliability Analysis Method (FORM) originates from structure and risk analysis [19] and was also used in various hydraulic engineering fields [12, 13]. It was adapted to the needs in morphodynamic simulations. The FORM applied to morphodynamic simulation to compute confidence limits of the evolution E of the river bottom with respect to one uncertain input parameter ω consists of three steps:

1. The statistical distribution of the uncertain parameter ω must be known or must be assumed. Only a Gaussian distribution is valid here. For a complete description of the Gaussian distribution, the mean value $<\omega>$ and the standard deviation σ_ω of the uncertain parameter ω must be specified.

2. The sensitivities, which are the partial derivatives $\partial E/\partial\omega$ of the state variable E with respect to the uncertain parameter ω, must be computed at the evaluation point $\omega = <\omega>$. Multiplying these sensitivities with the standard deviation σ_ω of the uncertain parameter ω gives the standard deviation σE_ω of the state variable:

$$\sigma E_\omega = \sigma_\omega \cdot \left.\frac{\partial E}{\partial \omega}\right|_{\omega=<\omega>} \tag{4}$$

3. From the standard deviation σE_ω the confidence limits connected to a given confidence level α can be computed. For normally distributed ω and a confidence level of $\alpha = 95\%$ the confidence limits are given by $CL_{\min} = E - 2 \cdot \sigma E_\omega$ and $CL_{\max} = E + 2 \cdot \sigma E_\omega$. Consequently, the confidence interval for E has a width of $4 \cdot \sigma E_\omega$.

The evolution of the river bottom E is the only state variable in our setup. Several input parameters are considered as uncertain: friction coefficient ks, mean grain size d_m, slope coefficient β, and others.

The computation of independent confidence limits for a set of uncertain parameters requires a separate application of the FORM or of the MCCL method for each parameter. In nearly all cases the number of uncertain parameters is orders of magnitudes smaller than the number of simulations required by the MCCL method. If compound confidence levels for a set of uncertain parameters are desired a single application of the FORM or of the MCCL method is sufficient. However, every application of the FORM requires one model evaluation with sensitivities in contrast to N model runs for the MCCL method.

3 AUTOMATIC SENSITIVITIES FOR SISYPHE

Algorithmic differentiation (AD) [7] is a method for computing derivatives of functions implemented as numerical simulation programs. Refer to [2, 3, 4, 5] for an impressive collection of successful applications of AD to a wide variety of real-world applications. Information on tools, publications, and applications can be obtained from the communities web portal

`www.autodiff.org` .

In this paper we focus on to the so-called tangent-linear or forward mode of AD as in our morphodynamic simulations the number of input variables is relatively small compared to the number of output variables. In optimisation scenarios the situation is different: Many input variables are mapped into only a few (or a single) output value(s), and sensitivities (gradients or even higher-order derivatives) of the outputs with respect to a potentially very large number of inputs are required. To compute these derivatives efficiently the adjoint or reverse mode of AD is required. See, for example, [7] for details.

3.1 Tangent-Linear Mode AD

A (Fortran) code is considered as an implementation of a nonlinear multivariate vector function

$$\mathbf{y} = F(\mathbf{x}, \mathbf{z}), \quad F : I\!R^{n+\tilde{n}} \to I\!R^m. \tag{5}$$

The goal is to compute directional derivatives ("tangents") of $F(\mathbf{x}, \mathbf{z})$, that is, products of the Jacobian matrix

$$F' = F'(\mathbf{x}, \mathbf{z}) \equiv \left(\frac{\partial y_j}{\partial x_i} \right)_{i=1,\ldots,n}^{j=1,\ldots,m} \tag{6}$$

containing the sensitivities (partial derivatives) of all *active* outputs (or *dependent* variables) $\mathbf{y} = (y_1, \ldots, y_m)^T$ with respect to the active inputs $\mathbf{x} = (x_1, \ldots, x_n)^T$ (or *independent* variables) with a vector $\dot{\mathbf{x}} \in I\!R^n$. The vector $\mathbf{z} \in I\!R^{\tilde{n}}$ contains all passive inputs, that is, variables that $\mathbf{y}$ depends on but whose impact on $\mathbf{y}$ is of no interest in the given context. We are looking for ways to evaluate the *tangent-linear model (TLM)* of $\mathbf{y} = F(\mathbf{x})$ defined as

$$\dot{\mathbf{y}} = \dot{F}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{z}) \equiv F'(\mathbf{x}, \mathbf{z}) \cdot \dot{\mathbf{x}}, \quad \dot{\mathbf{y}} \in I\!R^m \quad . \tag{7}$$

Note that the TLM is evaluated numerically at the evaluation point $(\mathbf{x}, \mathbf{z})$ specified by the values of the input variables of the model.[3]

Finite difference quotients (FD) can be used to compute an approximation of the TLM as follows:

$$\dot{\mathbf{y}} \approx \frac{F(\mathbf{x} + h \cdot \dot{\mathbf{x}}, \mathbf{z}) - F(\mathbf{x}, \mathbf{z})}{h} \quad . \tag{8}$$

The quality of this approximation depends to a large extend on the user's ability to pick the "right" value for h. For complex simulations this search amounts to trial and error. Moreover the objective remains unclear since the quality of a given approximation cannot be evaluated without knowledge of exact values. Trusting FD approximations of derivatives can be dangerous. Refer, for example, to [9] for further discussion of this method. An impressive illustration of the harm inflicted by approximate derivatives in the context of a matrix-free Truncated Newton algorithm [6] for unconstrained nonlinear optimisation can be found in [15]. Sensitivities by AD are exact up to machine precision. No truncation errors are introduced.

Conceptually, AD considers the function F as a decomposition into a sequence of p elementary assignments

$$\begin{aligned} &\text{for } j = n+\tilde{n}+1, \ldots, n+\tilde{n}+p \\ &\qquad v_j = \varphi_j(v_i)_{i \prec j} \end{aligned} \tag{9}$$

evaluating a single basic arithmetic operation $(+, -, *, \ldots)$ or intrinsic function (sin, cos, exp, ...). Equation (9) is also referred to as the *code list* of the given implementation of F. The relation $i \prec j$ denotes a direct dependence of v_j on v_i as an argument of φ_j. The TLM in Equation (7) is obtained by augmenting the code list of F with statements for propagating the vector $\dot{\mathbf{x}}$ through the code list (after initialising $\dot{v}_j = \dot{x}_j, \ j = 1, \ldots, n$):

$$\begin{aligned} &\text{for } j = n+\tilde{n}+1, \ldots, n+\tilde{n}+p \\ &\qquad \dot{v}_j = \sum_{i \prec j} \frac{\partial \varphi_j}{\partial v_i}(v_k)_{k \prec j} \cdot \dot{v}_i \\ &\qquad v_j = \varphi_j(v_i)_{i \prec j} \quad . \end{aligned} \tag{10}$$

[3] AD computes sensitivities numerically at the runtime of the differentiated code. No symbolic differentiation is performed. The function F must not be known explicitely, it might be given as an numerical approximation code only. In case of the morphodynamic model discussed in the present paper the function F is a bedload transport equation solved by a Finite Element Method implemented in Sisyphe.

1	$v_1 = x$	$\dot{v}_1 = \dot{x}$
2	$v_2 = z$	$\dot{v}_2 = 0$
3	$v_3 = v_1 + v_2$	$\dot{v}_3 = \dot{v}_1 + \dot{v}_2$
4	$v_4 = \sin(v_3)$	$\dot{v}_4 = \cos(v_3) \cdot \dot{v}_3$
5	$v_5 = \cos(v_1)$	$\dot{v}_5 = -\sin(v_1) \cdot \dot{v}_1$
6	$v_6 = v_4 \cdot v_5$	$\dot{v}_6 = v_4 \cdot \dot{v}_5 + \dot{v}_4 \cdot v_5$
7	$y = v_6$	$\dot{y} = \dot{v}_6$

Table 1: Code list (column 2) and tangent-linear code list (columns 2,3) for the example function in Equation (11).

The partial derivatives of the elemental functions φ_j are given by the well-known differentiation rules. Moreover, the computation of $\dot{v}_j$ as a function of $\dot{v}_i$ exploits the chain rule of differential calculus. Griewank and Walther [7] show that the computational cost $W(\dot{F})$ of evaluating a TLM of F is bounded by $W(\dot{F}) = c \cdot W(F)$ with $c \in [2, 5/2]$, where $W(F)$ denotes the costs of evaluating F. Moreover a TLM of F will allocate roughly twice the memory occupied by F.

To illustrate tangent-linear mode AD a simple example is considered. Let F be given as

$$y = F(x, z) = sin(x + z) * cos(x) \tag{11}$$

with active input $x \in I\!R$, passive input $z \in I\!R$, and active output $y \in I\!R$. The second column of Table 1 represents the code list of a possible implementation of Equation (11). The code list in Equation (9) is surrounded by code representing the initialisation with the input values (rows 1,2), and the extraction of the output value y (row 7). The third column of Table 1 contains the additional code propagating the directional derivative of the input $\dot{x}$ towards the desired directional derivative $\dot{y}$ of the output variable y.

The initial assignment of $\dot{\mathbf{x}}$ to their representatives $\dot{v}_i$, $i = 1, \ldots, n$ is called *seeding*, whereas the retrieval of the directional derivatives of the outputs from the tangent-linear code list is often referred to as *harvesting*.

3.2 The Differentiation-Enabled NAG Fortran Compiler

The differentiation-enabled NAG Fortran compiler [16] (from now on simply referred to as "the compiler") is developed as a joint effort of RWTH Aachen University, the University of Hertfordshire, and the Numerical Algorithm Group Ltd. The compiler generates tangent-linear and adjoint versions of Fortran codes. In the compiler's internal representation all floating-point variables are replaced by a derived data-type `compad_type`. The compiler provides a set of modules (`compad_module`'s) that define several `compad_type`'s. The AD-mode of the compiler (overloaded tangent-linear, tangent-linear, adjoint, second-order adjoint, tape based adjoints, ...) is chosen by command line options, which select a specific `compad_module`. Every `compad_module` provides overloaded operators and intrinsic functions implementing the AD functionality for its own `compad_type` as well as

combinations of the `compad_type` with the various intrinsic data types such as real or integer.

The TLM discussed in this paper is based on the `compad_type` defined in Figure 3a that combines the value (component `val` $\equiv v_i$) with its directional derivative (component `drv` $\equiv \dot{v}_i$) in one entity. The corresponding module `compad_scalar_module` defines overloaded operators and intrinsic functions that combine both steps of the TLM defined in Equation (10). As an example, the code fragment in Figure 3b shows the implementation of the tangent-linear sine function. It computes the value `r%val` of the output variable `r` (line 5) alongside with the propagation of the directional derivative `x%drv` of the input variable `x` towards the directional derivative `r%drv` of the output (line 4). Therefore `x%drv` is multiplied by the partial derivative of sin evaluated at `x%val`, that is, cos(`x%val`).

```
1 TYPE compad_type
2   SEQUENCE
3   DOUBLE PRECISION :: val = 0.D0
4   DOUBLE PRECISION :: drv = 0.D0
5 END TYPE compad_type
```

(a) Definition of `compad_type`

```
1 ELEMENTAL FUNCTION sin_ct(x) RESULT(r)
2   TYPE(compad_type), INTENT(in) :: x
3   TYPE(compad_type)             :: r
4   r%drv = x%drv * COS( x%val )
5   r%val = SIN( x%val )
6 END FUNCTION sin_ct
```

(b) Overloaded sine function

```
1 SUBROUTINE f(x,z,y)
2   DOUBLE PRECISION   :: x, z
3   DOUBLE PRECISION   :: y
4
5   y = SIN(x+z) * COS(x)
6
7 END SUBROUTINE f
```

(c) Code to be differentiated

```
1 PROGRAM p
2   USE compad_scalar_module
3   IMPLICIT NONE
4   TYPE(compad_type)   :: x, z, y
5   x = 1.1D0  ; z = 3.D0
6   CALL SEED( x, 1 )
7   CALL F( x, z, y )
8   PRINT *,'V:',VALUE(y),'␣D:',DERIV(y)
9 END PROGRAM p
```

(d) Active driver program

Figure 3: Example codes

Figure 3c shows a possible Fortran implementation of the simple example in Equation (11). This code gets differentiated in overloaded tangent-linear mode automatically by compiling with the options `-AD_TLM_SCALAR`, and `-AD_OVERLOAD`. In the resulting object code the variables `x`, `z`, and `y` are of type `compad_type` (Figure 3a). The operators (`*`, `+`, assignment) and intrinsic functions (`sin`, `cos`) in line 5 are resolved during the (AD-enabled) compilation to overloaded versions defined in `compad_scalar_module`.

A driver program (Figure 3d) calls the differentiated code. Therefore the module `compad_scalar_module` is included (line 2), and variables are declared as `compad_type` (line 4). Assignments (line 5) are overloaded: The value on the right hand side is assigned to the `val`-component of the `compad_type` variable on the left, and the `drv`-component is set to `0`. The directional derivative of the active input `x` is seeded by calling a subroutine from the `compad_scalar_module` (line 6) that assigns the value of the second argument (`1`) to the `drv`-component of the first argument. Calling the differentiated code (line 7)

is followed by diagnostic output.

The driver program must be compiled and linked with the additional option `-AD_DRIVER` to get access to the contents of `compad_scalar_module`. An executable is build by linking driver, differentiated code, and `compad_scalar_module`. Execution results in the value $F(x,z)$ and its derivative $dF(x,z)/dx$ at point $(x,z)^T = (1.1,3)^T$ printed onto the screen:

```
V:  -0.3711673238301428  D:   0.4685166713003768
```

Overloaded operators in the TLM introduce a significant runtime overhead: In the original model arithmetic operations and intrinsic functions are handled directly within the processor by operations coded in hardware. Replacing these intrinsic operations by overloaded operators introduces subroutine calls. Moreover intermediate results that are needed in subsequent operations (Table 1: v_3 used by v_4) might be held within processor registers for the intrinsic operations, whereas the arguments for overloaded operators are stored on the stack. Thus the theoretical slowdown factor $c \in [2, 5/2]$ (see Section 3.1) introduced by the additional computation of directional derivatives turns out to be higher if overloaded operators are used to implement the TLM. We observe a factor of roughly 5 between the runtime of the original model and the associated overloaded TLM.

The overhead of calling overloaded operators can be reduced by a second transformation stage in the compiler: Overloaded operators from the `compad_module` are replaced by code that implements the differentiation rules directly within the internal representation of the compiler. Thereby the need of calling a subroutine for every arithmetic operation is removed and the slowdown factor can be reduced to its theoretical limit.

3.3 Tangent-Linear Model of Sisyphe

The two-dimensional morphodynamic model Sisyphe [18] consists of roughly 100000 lines of Fortran code. Differentiating the code with the compiler yields a dramatic change in meaning and content of the code: The data type of variables, constants, and arguments of subprograms might be changed. New variables, arguments, or even new subprograms are created. Operators and intrinsic functions are replaced by overloaded versions. Moreover, our compiler introduces a large number of identifiers by including a `compad_module`. Name clashes[4] and other ambiguities might require intervention by hand.

Another more serious problem arises from changing the data type of floating-point variables to `compad_type`: In the differentiated version of the code any data transfer statement (`READ`, `WRITE`, `PRINT`) with a variable of type `compad_type` will transfer twice as much data (two components of `compad_type`) as in the non-differentiated version. Hence, inputs files are invalidated and the structure of output files is changed. The compiler offers the command line option `-AD_DEACTIVATE_IO` to restrict `compad_type` variables in

[4]For example, in Fortran there is no problem in naming a variable `sum`. However, inclusion of a `compad_module` will introduce an overloaded version of the intrinsic function `sum` for vectors of `compad_type` giving `sum` a second, conflicting meaning.

I/O statements to their value component automatically. It also provides a set of macros for accessing the components of `compad_type`. Those have to be incorporated into the source code by hand if data transfer is executed by external libraries (for example, NETCDF[5]) or if the program can restart an interrupted program execution from previously stored snapshots. To ensure correctness of the tangent-linear code snapshots must also contain directional derivatives.

3.4 Tangent-Linear Model of Sisyphe and First-Order Reliability Analysis

Sisyphe computes the evolution $E \in I\!R^m$ of the river bottom on a grid of m nodes from a vector $\mathbf{x}$ of n uncertain parameters and a vector $\mathbf{z}$ of $\tilde{n}$ other inputs:

$$E = E(\mathbf{x}, \mathbf{z}) = \begin{pmatrix} E_1(\mathbf{x}, \mathbf{z}) \\ E_2(\mathbf{x}, \mathbf{z}) \\ \vdots \\ E_m(\mathbf{x}, \mathbf{z}) \end{pmatrix} \quad , \quad E : I\!R^{n+\tilde{n}} \rightarrow I\!R^m \quad . \tag{12}$$

In Section 4.1 four input parameters are assumed to be uncertain: parameter for secondary currents α, slope coefficient β, friction coefficient ks, and the mean grain size d_m.[6] Hence,

$$\mathbf{x} = (\alpha, \beta, d_m, ks)^T \in I\!R^4 \quad . \tag{13}$$

Computing confidence limits of E with respect to a specific uncertain parameter, for instance α, by the FORM requires to compute the standard deviation

$$\sigma E_\alpha = \sigma_\alpha \cdot \left. \frac{\partial E}{\partial \alpha} \right|_{\alpha = <\alpha>} \tag{14}$$

of E with respect to α (see Section 2.2, Equation (4)). Therefore the standard deviation σ_α of α is multiplied with the partial derivatives of E with respect to α at the evaluation point $\alpha = <\alpha>$. Both mean value $<\alpha>$ and standard deviation σ_α of the uncertain parameter must be given (Section 4.1, Table 2).

The generated TLM of Sisyphe computes products of the Jacobian matrix of the bottom evolution $E = E(\mathbf{x}, \mathbf{z}) = E(\alpha, \beta, d_m, ks, \mathbf{z})$ with a seed vector $\dot{\mathbf{x}} = \left(\dot{\alpha}, \dot{\beta}, \dot{d}_m, \dot{ks}\right)^T$:

$$\dot{E}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{z}) = E'(\mathbf{x}, \mathbf{z}) \cdot \dot{\mathbf{x}} = \begin{pmatrix} \frac{\partial E_1(\mathbf{x},\mathbf{z})}{\partial \alpha} & \frac{\partial E_1(\mathbf{x},\mathbf{z})}{\partial \beta} & \frac{\partial E_1(\mathbf{x},\mathbf{z})}{\partial d_m} & \frac{\partial E_1(\mathbf{x},\mathbf{z})}{\partial ks} \\ \frac{\partial E_2(\mathbf{x},\mathbf{z})}{\partial \alpha} & \frac{\partial E_2(\mathbf{x},\mathbf{z})}{\partial \beta} & \frac{\partial E_2(\mathbf{x},\mathbf{z})}{\partial d_m} & \frac{\partial E_2(\mathbf{x},\mathbf{z})}{\partial ks} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial E_m(\mathbf{x},\mathbf{z})}{\partial \alpha} & \frac{\partial E_m(\mathbf{x},\mathbf{z})}{\partial \beta} & \frac{\partial E_m(\mathbf{x},\mathbf{z})}{\partial d_m} & \frac{\partial E_m(\mathbf{x},\mathbf{z})}{\partial ks} \end{pmatrix} \begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{d}_m \\ \dot{ks} \end{pmatrix} \quad . \tag{15}$$

[5] `www.unidata.ucar.edu/software/netcdf/`

[6] The letter m in d_m does not refer to the dimension of E.

Hence evaluating the TLM of Sisyphe (Equation (15)) with $\mathbf{x}_\alpha = (<\alpha>, \beta, d_m, ks)^T$ and seed vector $\dot{\mathbf{x}}_\alpha = (\sigma_\alpha, 0, 0, 0)^T$ computes the desired standard deviation (Equation (14)) of E with respect to the uncertain parameter α:

$$\sigma E_\alpha = \sigma_\alpha \cdot \left.\frac{\partial E}{\partial \alpha}\right|_{\alpha=<\alpha>} = \dot{E}(\mathbf{x}_\alpha, \dot{\mathbf{x}}_\alpha, \mathbf{z}) \quad . \tag{16}$$

Computing standard deviations of E with respect to other uncertain inputs require additional evaluations of the TLM of Sisyphe with $\mathbf{x}$ and $\dot{\mathbf{x}}$ set accordingly.

Compound confidence levels for a set of uncertain parameters are obtained by specifying mean values and standard deviations for all desired uncertain inputs. In Section 4.1 the compound dependency of the bottom evolution E with respect to all four uncertain inputs is computed by evaluating the TLM of Sisyphe (Equation (15)) with

$$\mathbf{x}^T = (<\alpha>, <\beta>, <d_m>, <ks>) \quad \text{and} \quad \dot{\mathbf{x}}^T = (\sigma_\alpha, \sigma_\beta, \sigma_{d_m}, \sigma_{ks}) \quad . \tag{17}$$

4 APPLICATIONS AND RESULTS

4.1 Laboratory Experiment with 180° Bend

In a hydraulic model of a 180° bend a flat bottom was developed due to a 5 hours hydrograph (see Figure 4, and [20] for further details). The underlying discretisation consists of more than 1800 nodes. This experiment is used to validate the sediment transport module Sisyphe with respect to the effect of secondary currents. The results of the coupled hydrodynamic / morphodynamic model are reasonably satisfying for a depth averaged model (see Figure 5).

The FORM is applied to compute confidence limits of the bottom evolution concerning the following four parameters: parameter for secondary currents α, slope coefficient β, friction coefficient ks, and the mean grain size d_m. The sensitivities of the bottom evolution E with respect to each uncertain input parameter required by the FORM are calculated with the differentiated version of Sisyphe.

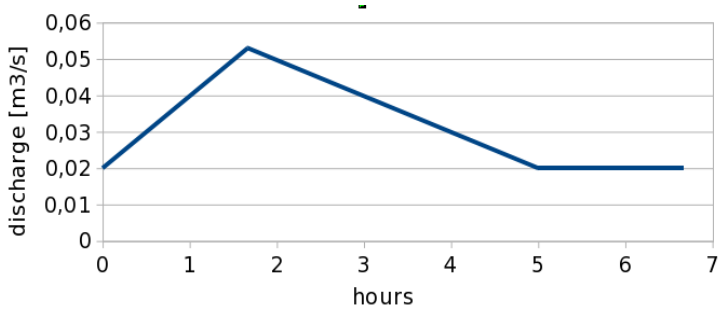


Figure 4: Hydrograph (5 hours) of the 180° bend experiment.

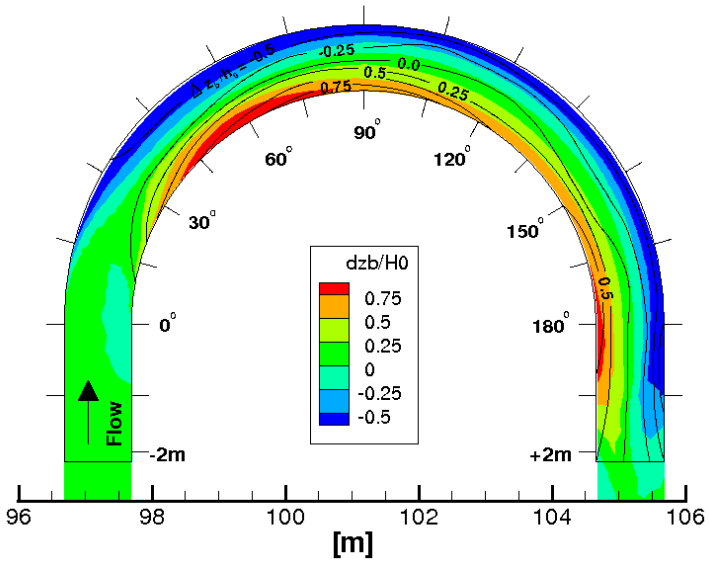


Figure 5: Comparison of the normalised bottom evolution from the physical model (black isolines) and the numerical model (coloured areas) for the 180° bend experiment.

The four input parameters are defined independently as uncertain by assigning the corresponding mean value from Table 2 as their initial value together with seeding their sensitivity component with the standard deviation given in Table 2 (see Section 3.4 for details). Figure 6 shows the standard deviations σE_{d_m}, σE_{ks}, σE_β, and σE_α of the bottom

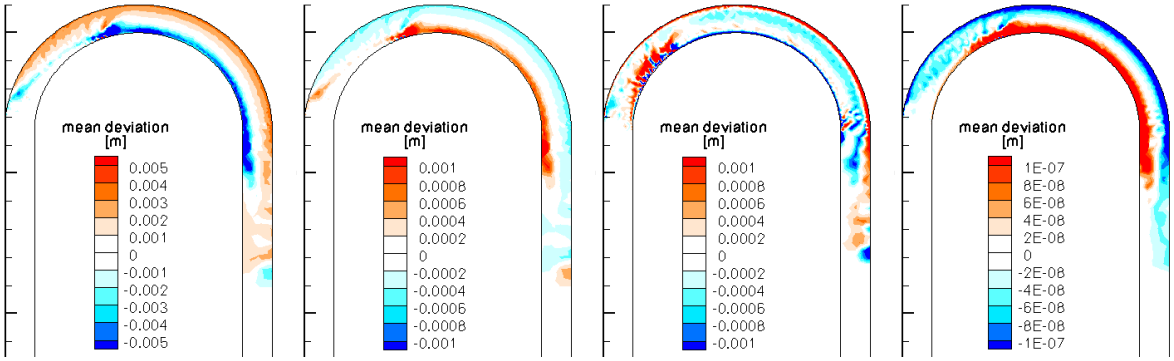
Input parameter	**Mean value**	**Standard deviation**
parameter of secondary currents α	7.0	1.0
slope coefficient β	1.3	0.4
friction coefficient ks	3.0 mm	0.1 mm
mean grain size d_m	1.0 mm	0.1 mm

Table 2: Mean values and standard deviation for the four uncertain input parameters used in the 180° bend experiment.

evolution E with respect to the uncertain input parameters d_m, ks, β, and α respectively. For the chosen configuration the bottom evolution E is most sensitive to the mean grain size d_m (Figure 6a). The impact of the slope coefficient β and of the friction coefficient ks is 5 times smaller (Figures 6b and 6c). The coefficient for the effect of secondary currents α has a surprisingly small influence (Figure 6d) on the bottom evolution E.

A typical cross section in a bend has steep water at the outer bend and shallow water at the inner bend. An increased friction coefficient or coefficient of secondary currents increase the slope in the cross sections. Note the negative values at the outer bend and

positive values at the inner bend in Figures 6b and 6d. In contrast an increased mean grain size or slope effect will decrease the slope effect of the cross sections. Note the positive values at the outer bend and negative values at the inner bend in Figures 6a and 6c. After 5 hours the computed bottom evolution shows 3 cm erosion at the outer



(a) (b) (c) (d)

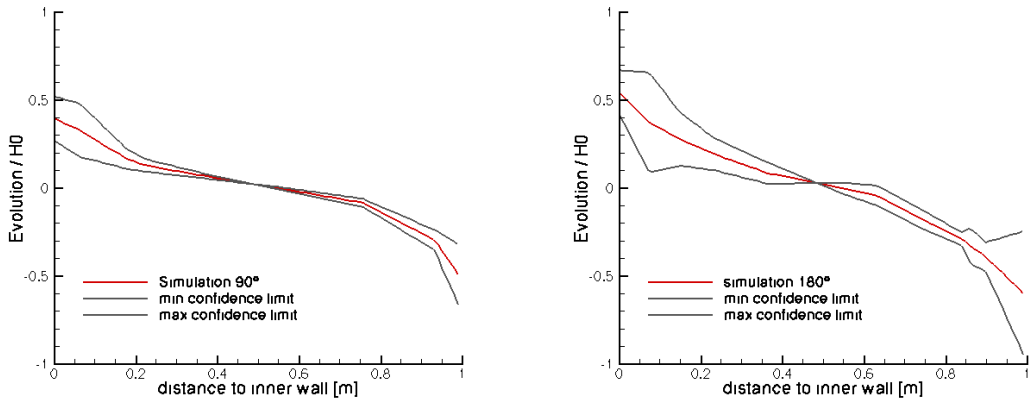
Figure 6: Standard deviation of the bottom evolution E in the 180° bend experiment considering a Gaussian distribution of the (a) mean grain size d_m (with a standard deviation of 0.1 mm), (b) friction coefficient ks (0.1 mm), (c) slope coefficient β (0.1 mm), and (d) parameter of second currents (0.1 mm).

bank and 3 cm sedimentation at the inner bank. The calculated sensitivities are relatively large. The 95 % confidence limits are twice the value of the standard deviation. Hence a confidence interval of ±1.5 cm is found, which is half of the maximum bottom evolution value.

The compound dependency of the bottom evolution E with respect to all 4 uncertain input parameters α, β, ks, and d_m are calculated by an additional application of the FORM. The compound 95% confidence limit for the bottom evolution E (see Figure 7) at the 90° cross section is significantly smaller than that at 180°. Moreover, at the inner part of the channel the simulation is far less uncertain than at the boundaries.

4.1.1 Comparison with the MCCL Method

The results of the FORM obtained by using exact sensitivities provided by a tangent-linear version of Sisyphe are validated against the MCCL method with 100 simulation runs. Figure 8 shows the compound 95% confidence interval of the bottom evolution E considering the four uncertain parameters α, β, ks, and d_m. While both methods show a good qualitative match the quantitative information differ significantly. In particular, the FORM shows higher uncertainties (factor 4) in the resulting bottom evolution at the boundaries. The reasons remain unclear so far. Other models have shown much better matching results ([10], [11]). AD allows to compute local sensitivities at the evaluation



(a) cross section 90° (b) cross section 180°

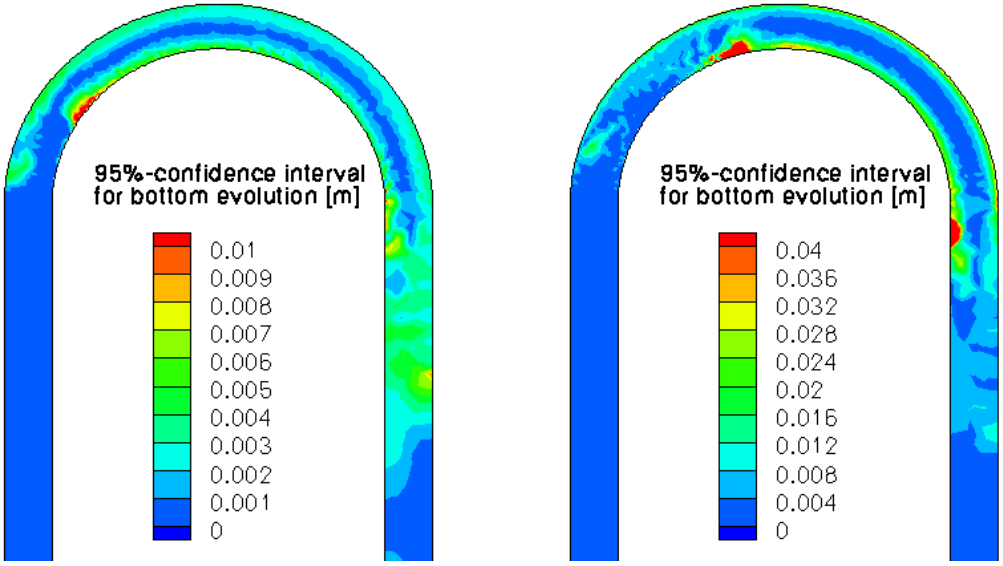
Figure 7: Simulated normalised evolution (red line) at cross section 90° (a) and at 180° (b) for the 180° bend experiment. The grey lines mark the compound 95% confidence limits for the following 4 uncertain input parameters: parameter for secondary currents α, slope coefficient β, friction coefficient ks, and the mean grain size d_m.

point with machine accuracy. With the standard deviation specified in Table 2 the FORM computes confidence limits for a rather large interval by multiplying the obtained sensitivity information with the standard deviation of the uncertain parameter. This introduces a so far unspecified approximation error. On the other hand, Monte Carlo methods are approximation processes. They have a smoothing effect on the approximated function. Evaluation of the model at selected points might miss structurally important areas.

Until now TLM-Sisyphe, the tangent-linear version of Sisyphe, cannot be coupled with Telemac-2D directly as no tangent-linear version of Telemac-2D is currently available[7]. As a workaround the hydrodynamics required by TLM-Sisyphe is pre-calculated by a coupled run of the Sisyphe and Telemac-2D and is written to disc every 180 seconds of simulation time. The morphodynamics computed by TLM-Sisyphe within the FORM requires a ten times smaller time step than the hydrodynamics. Thus every hydrodynamic state read from disc is followed by the extrapolation of the nine missing states. This approximation of the hydrodynamics might be the main reason for the observed discrepancies between the compound 95% confidence limits obtained by the FORM and the MCCL method. A deeper investigation of the effect of the hydrodynamic interpolation is subject of ongoing work.

In numerical morphodynamic modelling the computing time spent for evaluating the models is crucial due to long-term large-scale questions. A single evaluation of a model can easily take days or weeks. In the 180° bend experiment 100 simulation runs can be

[7]The development of TLM-Telemac-2D is the subject of a new project starting in summer 2010.



(a) Confidence interval by MCCL (b) Confidence interval by FORM

Figure 8: Compound 95% confidence interval of the bottom evolution E considering four uncertain input parameters α, β, ks, and d_m for the 180° bend experiment calculated with MCCL (a) and FORM (b).

performed for the MCCL method within 200 minutes. The subsequent statistical interpretation of the results takes an additional 100 minutes. The code is compiled using the highly optimising Intel Fortran compiler (Options used: `-O3 -ftz -fno-alias -align -override_limits`; Compilation with `-O4` took too much time).

The runtime of the FORM is clearly dominated by the evaluation of the TLM generated by the differentiation-enabled NAG Fortran compiler (Options used: `-O4`). 22 minutes are spent overall to compute the confidence intervals with the FORM, which amounts to only 7.33% of the runtime of the MCCL method. Moreover, the differentiation-enabled NAG Fortran compiler currently uses the GNU `gcc` compiler to generated binary code. On our target architectures gcc's optimisation capabilities turn our to lack behind those of the Intel compiler suite.

The evaluation of the original model with the NAG Fortran compiler required 4:15 minutes. Consequently, the TLM with overloaded operators suffers from a slowdown factor of approximately 5.2 with respect to the original model.

4.2 Mühlham Bend (River Danube)

A computational more complex test case is a 10 km long stretch of the river Danube including a 270° bend (see Figure 9). Nearly 10^5 grid elements are used for the discretization with mean node distances of about 6 m in the river channel and up to 30 m at the

flood planes.

Figure 9: River Danube model area with a 270 ° bend of Mühlham.

The following 10 parameters are assumed to be uncertain: active layer thickness AL, coefficient for the slope effect β, the coefficient for the bedload formulation of Meyer-Peter & Müller MPM, coefficient for the secondary current effect α, friction coefficient ks, and the grain sizes d_{m1} .. d_{m5} of 5 classes. See Table 3 for their values and standard deviation, again assuming a Gaussian distribution.

In the FORM of this complex system a direct or at least frequent coupling between hydrodynamics and morphodynamics is essential for simulating a natural hydrograph. The large number of elements in this model makes it impossible to precompute and store on disc a sufficient number of hydrodynamic states by the coupled non-AD version of Sisyphe and Telemac-2D. Thus TLM-Sisyphe is not running with a natural hydrograph, but with steady state conditions at a mean discharge of 650 m^3/s. For this discharge a mean bottom evolution is expected, so that a coupling to the hydrodynamics could be avoided. The continuity equation is solved within the TLM-Sisyphe version. In the literature [18] it is assumed that solving the motion equation by Telemac-2D is required if the bottom evolution reaches more than 10% of the water depth. Unfortunately, this state is reached after 12 hours simulation time. So the reliability analysis could only applied for this short period, because TLM-Telemac-2D is not yet available (see Section 4.1.1).

Nevertheless the result of the compound 95 % confidence interval of bottom evolution concerning all 10 uncertain parameters (see Figure 10a) shows the same qualitative behaviour as a reliability analysis performed with the MCCL method on a 9-day artificial hydrograph with two high water periods each day (see Figure 10b). The quantitative results are not comparable due to the following three aspects:

- The different hydrodynamic situation: 12 hours, uncoupled hydrodynamic and morphodynamic, steady state conditions (FORM) in contrast to 9 days, coupled hydro-

Input parameter	Mean value	Standard deviation
active layer thickness AL	0.1	0.02
slope coefficient β	1.3	0.4
MPM factor	4.0	0.3
parameter of secondary currents α	10.0	3.0
friction coefficient ks	50.0 mm	10.0 mm
mean grain size d_{m1}	0.5 mm	0.1 mm
mean grain size d_{m2}	2.5 mm	0.3 mm
mean grain size d_{m3}	10.0 mm	1.0 mm
mean grain size d_{m4}	24.0 mm	4.0 mm
mean grain size d_{m5}	48.0 mm	4.0 mm

Table 3: Mean values and standard deviation for the chosen uncertain input parameters in Mühlham bend experiment (river Danube).

dynamic and morphodynamic, artificial hydrograph with two high water periods per day (MCCL).

- In the MCCL method each area with the same friction coefficient is handled as an extra parameter. So instead of 1 uncertain friction coefficient ks in case of the FORM there 4 different friction coefficients $ks_{\{1,\ldots,4\}}$ are taken into account for the river channel, the groynes, an area with a nature experiment and a sandbank with trees with the MCCL method.

- Furthermore, the chosen probability distribution is in most cases double Gaussian and not a simple Gaussian distribution.

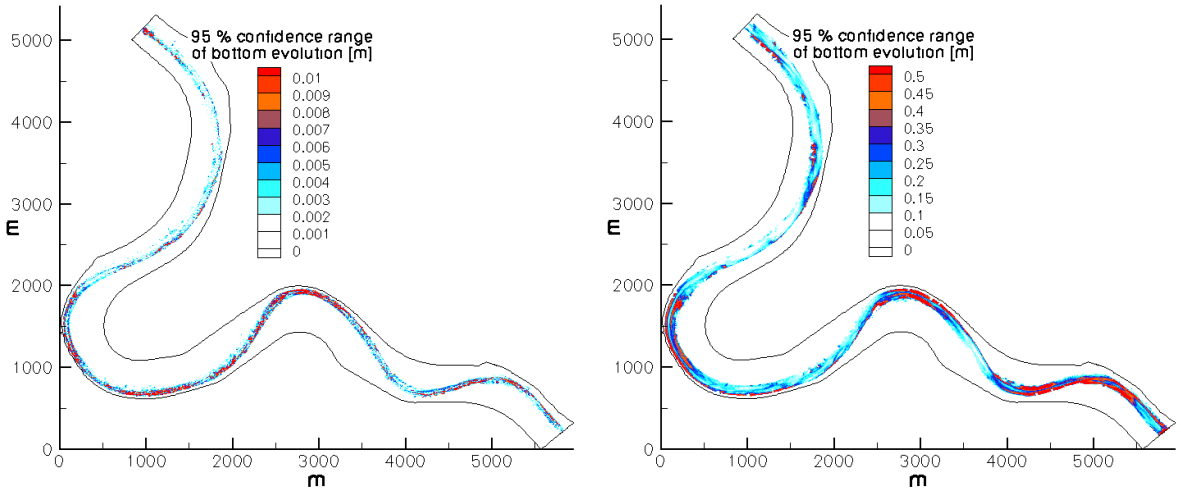
As expected mainly from the different hydrodynamics the bottom evolution deviation is higher in case of the MCCL method than in the FORM. There is no discussion of the consumed computing time due the different simulation time horizons (12 hours and 9 days).

5 CONCLUSIONS AND OUTLOOK

We have computed confidence limits of the bottom evolution E with respect to several uncertain input parameters by the First-Order Reliability Analysis Method at a fraction of the computational cost of the Monte Carlo Confidence Limit method.

A good qualitative match between results of the FORM and the MCCL method was observed. Substantial differences in the underlying evaluation scenarios might be responsible for the qualitative differences observed:

- TLM-Sisyphe morphodynamics is not coupled to Telemac-2D hydrodynamics, since there is no TLM of Telemac-2D so far. By interpolating the missing hydrodynamic information from certain precomputed stages an additional approximation error is



(a) Compound 95% confidence interval, FORM (b) Compound 95% confidence interval, MCCL

Figure 10: Compound 95% confidence interval of bottom evolution of the Mühlham experiment (a) concerning 10 uncertain input parameters after 12 h mean discharge computed by the FORM, and (b) 13 uncertain input parameters after 9 days hydrograph obtained by the MCCL method.

introduced (180° bend, Section 4.1.1). Assuming steady state conditions for the hydrograph results in a different hydrodynamical scenario (Mühlham, Section 4.2).

- For the MCCL method four friction coefficients were used, instead of a uniform friction coefficient in the FORM (Mühlham, Section 4.2).

We are confident that once we can run identical experiments we will get comparable results by the FORM with much less computational effort than in the MCCL method.

On the other hand, Monte Carlo methods have a smoothing effect on the function they are approximating due to the fact that evaluations of the model are done only on some sample points. Rapid changes in small domains (local behaviour of the model) or even discontinuities may not be captured. This is the price to pay for approximating the behaviour of the function in an interval $(\omega - \sigma_\omega, \omega + \sigma_\omega)$ instead of computing a value for a single evaluation point.

In contrast computing sensitivities with AD is a strictly local approach: The sensitivities by AD are exact up to machine precision at the current point. For nearly linear functions in the desired input parameters the sensitivities computed by AD will be good even over larger intervals. But for strongly non-linear models the sensitivities might change even in small neighbourhoods of the current evaluation point significantly. Thus the confidence level computed within FORM by multiplying the standard deviation of the uncertain input by the sensitivity of the output (bottom evolution E here) with respect to the uncertain input is only an approximation.

It might be worth looking for clever combinations of the large interval covering Monte Carlo method (MCCL) with the locally exact information provided by the FORM based on exact sensitivities. Hopefully one can minimise the somehow contradicting approximation errors of both methods.

Furthermore, one might consider alternative methods for the propagation of uncertainties such as the methods of moments [1], or handling the uncertainties as genuine intervals instead of point wise. Switching to interval arithmetic will raise a number of new issues such as the need for special interval algorithms (for instance Interval-Newton [14] instead of Newton methods working on intervals directly).

Next steps will include a more detailed investigation of the error introduced by the hydrodynamic approximation, and the development of the differentiated Telemac-2d version.

REFERENCES

[1] M. Beckers and U. Naumann. Propagation of uncertainties using the method of moment. In *Proceedings of The Tenth International Conference on Computational Structures Technology*. Civil-Comp Press, 2010. Submitted.

[2] M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors. *Computational Differentiation: Techniques, Applications, and Tools*, Proceedings Series. SIAM, 1996.

[3] M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris, editors. *Automatic Differentiation: Applications, Theory, and Tools*, number 50 in Lecture Notes in Computational Science and Engineering, Berlin, 2005. Springer.

[4] G. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, editors. *Automatic Differentiation of Algorithms – From Simulation to Optimization*, New York, 2002. Springer.

[5] G. Corliss and A. Griewank, editors. *Automatic Differentiation: Theory, Implementation, and Application*, Proceedings Series. SIAM, 1991.

[6] R. Dembo and T. Steihaug. Truncated-Newton algorithms for large-scale optimization. *Math. Prog.*, 26:190–212, 1982.

[7] A. Griewank and A. Walther. *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation (Second Edition)*. SIAM, 2009.

[8] J.-M. Hervouet, 2007. Hydrodynamics of Free Surface Flows: Modelling with the Finite Element Method. *John Wiley & Sons* (2007)

[9] C. T. Kelley. *Solving Nonlinear Equations with Newton's Methods*. SIAM, Philadelphia, 2003.

[10] R. Kopmann, A. Schmidt. Reliability analysis of two-dimensional morphodynamic results. *Proceedings of International Conference on Fluvial Hydraulics*, Turkey (2008)

[11] R. Kopmann, A. Schmidt. Comparison of different reliability analysis methods for a 2D morphodynamic numerical model of River Danube *Proceedings of International Conference on Fluvial Hydraulics*, Braunschweig (2010) (submitted)

[12] C.S. Melching. 1992. An improved first-order reliability approach for assessing uncertainties in hydrologic modeling, *Journal of Hydrology*, 132, pp157-177.

[13] Melching, C.S. 1992. II. Improved First-Order Uncertainty Method for Water-Quality Modeling, *Journal of Environmental Engineering*, pp 791-805, Sept/Oct (1992)

[14] R.E. Moore, R.B. Kearfott, and M.J. Cloud. 2009. Introduction to Interval Analysis, SIAM, Philadelphia.

[15] U. Naumann, M. Maier, J. Riehme, and B. Christianson. Automatic first- and second-order adjoints for Truncated Newton. In M. Ganzha et al., editor, *Proceedings of IMCSIT'07: Workshop on Computer Aspects of Numerical Algorithms (CANA'07)*, pages 541–555. PTI, 2007.

[16] U. Naumann and J. Riehme. A differentiation-enabled Fortran 95 compiler. *ACM Transactions on Mathematical Software*, 31(4):458–474, 2005.

[17] L. Nikitina, I. Nikitin, T. Clees. 2009. Studie Zuverlässigkeitsanalyse morphodynamischer Modelle. Abschlussbericht zum Arbeitspaket (WP) 2, Fraunhofer Institut Algorithmen und Wissenschaftliches Rechnen (2009)

[18] K. Villaret. 2005. User Manual Sisyphe Release 5.5. HP-76/05/009/A, Department National Hydraulics and Environment Laboratory, Electricité de France (2005)

[19] B.C. Yen,S. Cheng, and C.S. Melching. 1986. First order reliability analysis, Stochastic and risk Analysis in hydraulic Engineering. *International Symposium on Stochastic Hydraulics 4*, Water Recources Publication, Littleton, Colorado (1984)

[20] C. Yen, K.T. Lee. Bed Topography and Sediment Sorting in Channel Bend with Unsteady Flow, *Journal of Hydraulic Engineering*, Vol.121, No. 8, (1995)