Environmental Modelling & Software 34 (2012) 19–29

Contents lists available at SciVerse ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

A Matlab software framework for dynamic model emulation

W. Tych[a,*], P.C. Young[a,b]

[a] Lancaster Environment Centre, Lancaster University, UK
[b] Fenner School of Environment and Society, Australian National University, Canberra, Australia

ARTICLE INFO

Article history:
Received 12 January 2011
Received in revised form
25 July 2011
Accepted 17 August 2011
Available online 19 September 2011

Keywords:
Emulation
DACE
DBM
Dominant modes
Transfer function
Matlab

ABSTRACT

The paper describes a software framework for implementing the main stages of the Data Based Mechanistic (DBM) modelling approach to the reduced order emulation (meta-modelling) of large dynamic system computer models, within the Matlab software environment. The framework exploits routines in the CAPTAIN Toolbox to identify and estimate transfer function models that reflect the dominant modes of the dynamic behaviour in the large model. This allows for the 'nominal emulation' and validation of the large model for a single, specified set of parameters; as well as 'stand-alone, full emulation' based on the construction and validation of hyper-dimensional maps between a user-specified range of large model parameters and the parameters of the associated, low order transfer function models. The software framework uses the multivariable structure constructs available within Matlab™ to form a small library of routines that will become part of the Captain Toolbox. The library is formed around special data structures that facilitate multivariable operations and visualisations which both enhance the efficiency of the emulation modelling analysis and the modeller's interaction with the process of emulation. The nature of the analysis is illustrated by a topical example concerned with the emulation of the OTIS computer simulation model for the transport and dispersion of solutes in a river system.

1. Introduction

The present paper describes the main aspects of a software platform for the development and implementation of simple models that are able to emulate the dynamic behaviour of large computer simulation models. In this introductory section, we provide a brief outline of large model emulation and discuss the need for a software environment that facilitates the development of dynamic emulation models. This provides the background for subsequent sections that describe the details of the emulation model development within the proposed software environment. The final section considers the emulation of the OTIS *Transient Storage* (TS) model for simulating solute transport and dispersion in river systems. This demonstrates the kind of results generated by the software system and shows how it is possible to closely emulate the dynamic behaviour of this large, distributed parameter computer model by a 'stand-alone', low order and computationally efficient emulation model.

1.1. Emulation modelling

The emulation of large dynamic simulation models (sometimes termed 'meta-modelling') is a procedure that identifies a simple, computationally efficient dynamic model whose response characteristics mimic those of the complex model as closely as possible. This low order dynamic emulation model can then replace the large model in computationally intensive exercises such as uncertainty and sensitivity analysis; or in tasks such as forecasting and control system design. This procedure can take various forms but, in the present paper, we consider the *Data Based Mechanistic* (DBM) modelling approach to emulation proposed by Young and Ratto (Young and Ratto, 2009, 2011) (hereafter referred to as YR), where the simple model is obtained by a statistical process of identification and estimation, based on data obtained from planned experiments performed on the large simulation model. As such, this approach can be considered within the context of the Design and Analysis of Computer Experiments (DACE: see e.g. (Santner et al., 2003; Bayarri et al., 2007; Young and Ratto, 2009, 2011; Young, 2011)).

Large model emulation is also related to the topic of 'model order reduction' that has received a lot of attention in the control and systems literature over the last few decades. Many different approaches have been suggested for model reduction, from simple

* Corresponding author. Tel.: +44 1524 593973.
E-mail addresses: w.tych@lancaster.ac.uk (W. Tych), p.young@lancaster.ac.uk (P.C. Young).

1364-8152/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2011.08.008

methods, such as the 'power reduction' approach of Liaw (1986), to more complex procedures, including 'proper orthogonal decomposition' and 'balanced model order reduction' (Moore, 1981; Willcox and Peraire, 2002). However, while emulation modelling might utilize approaches to reduced order modelling such as these, the context is quite different and the methodology that we propose for emulation is much more flexible. For instance, most of these methods are based on direct reduction of the large model equations to a simpler, low order form, and so require direct access to these equations; whereas, in our approach, it is only necessary to simulate the model within the DACE context. In addition, a reduced order model *that is able to explain the high order model with sufficient accuracy* (not all reduced order models do this) could be considered as a 'nominal' emulation model Young and Ratto (2011), in the sense that it emulates a large simulation model that has a given, fixed set of parameters. However, this contrasts with the 'full' emulation model considered by YR and in the present paper, which is intended to emulate the large model over a whole, user-specified, range of large model parameters.

Nominal emulation modelling exercises have been considered previously in an environmental context (see e.g. Young et al. (1996b); Young (1998); Lees et al. (2000); Young and Parkinson (2002)) and they can be considered as forerunners to full emulation modelling approach considered here. In fact, Lees et al. (2000) consider the same *Transient Storage* (TS) model for simulating solute transport and dispersion in river systems that is employed in our later illustrative example. However, they are concerned with nominal emulation based on the specific problem of matching the temporal moments of the large model response by a DBM model (in their case the *Aggregated Dead Zone* (ADZ) model), within the specific context of data from a tracer experiment in a river system, *where the upstream concentration profile (the input to the model) is fairly smooth*. In contrast to this, full model emulation requires that the large model response will be emulated for a wide range of inputs, so that the input perturbations need to excite the model sufficiently to ensure the full identifiability of the emulation model in this wider context. Thus, as we shall see later, while a first order ADZ model can emulate the large simulation model response for smooth inputs *between individual reaches*, it is not able to fully emulate the large model over many reaches and for a wider range of input perturbations.

Finally in relation to the nature of emulation models, it is important to note that a 'computationally simple' emulation model does not necessarily need to have less parameters than the large simulation model. For instance, in the large TS simulation model mentioned above, there are only five main parameters. However, because it is a distributed parameter model that is solved numerically, the dynamic order considered in ordinary difference equation terms is very high, since it is dependent of the temporal and spatial discretisation intervals, as well as the spatial extent of the model. As we shall see in the later illustrative example, therefore, although the 6th order differential equation emulation model has more fundamental parameters (twelve in contrast to the five main parameters in the TS model), the computational time is almost one thousand time smaller.

1.2. A software environment for large model emulation

The software platform for implementing the DBM modelling approach to full emulation in a systematic manner exploits the transfer function modelling routines in CAPTAIN Toolbox Pedregal et al. (2005); Young (2006); Young and Garnier (2006); Taylor et al. (2007) [1], as well as other computational facilities available in the Matlab/Simulink™ software environment. It is intended that this software platform will provide easily usable tools that can help the research community wishing to apply the DBM modelling approach to emulation more widely and to develop the procedure further. In effect, the software becomes a living theory that assists the emulation process in various ways, such as communicating the developments to others in a modelling team, allowing more easily for new applications and providing a vehicle for the development of new methods and techniques.

The package includes a Matlab-based framework for numerical experiments on the large computer simulation model; for the identification and estimation of transfer function models characterising the observed dominant modes of the large model; and for the implementation of the resulting fast emulation model using an estimated multivariable mapping between the large model parameters and the estimated transfer function parameters. Note that the statistical concepts of model 'identification' and 'estimation' are used here. This is because the identification of a parsimonious model structure is a most important aspect of the reduced order transfer function modelling and so it is best to separate this from the subsequent process of parameter estimation, which yields the values of the parameters that characterize this model structure and so define the TF model response characteristics.

The whole process of reduced order model emulation has been developed as part of the DBM approach to modelling environmental and other dynamic systems (Young, 1998), where it provides a bridge between high order dynamic computer models and low order, data based models (Young and Ratto, 2009). Also, in sympathy with the philosophy of DBM modelling, the mapping structure, as well as transfer function model identification and estimation procedures, normally require a satisfactory mechanistic interpretation of the model, so avoiding the use of 'black-box' representations, such as wavelets and principal components, that generally have no clear physical meaning. In order to enhance this aspect of the analysis, much attention in the software development has been directed to the visualisation of results using the advanced procedures available for this in Matlab.

2. Design and analysis of computer experiments (DACE)

Here, we consider a large computer model of a dynamic system (hereafter referred to simply as the 'large model') with parameters $\mathbf{X} \in \mathscr{R}^p$, where p is the number of the large model parameters. Following the notation in Young and Ratto 2011, we shall seek an emulator of the large model output(s) $y(t) = F(\mathbf{y}, \mathbf{u}, \mathbf{X})$ in the form of $y(t) = f(\mathbf{y}, \mathbf{u}, \boldsymbol{\theta}(\mathbf{X}))$, with both $F()$ and $f()$ dependent on $\mathbf{X}$, and where $\mathbf{u}$ is the vector of system input(s), $\mathbf{y}$ the vector of system output(s), and $f(.)$ is a low order dynamic model with parameters $\boldsymbol{\theta}(\mathbf{X})$ that are mapped functions of the large model parameter vector $\mathbf{X}$. The low order dynamic model, for the case of the example considered later in this paper, is the following single input, single output, continuous-time *Transfer Function* (TF) model of order (m, n) with time delay τ, normally specified by the triad $[nm\tau]$:

$$y(t) = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_m}{s^n + a_1 s^{n-1} + \cdots + a_n} u(t-\tau) \tag{1}$$

where s^r is the time derivative operator, $(s^r = \frac{d^r}{dt^r})$ and the parameter vector is $\theta = [a_1\ a_2 \ldots a_n\ b_0\ b_1 \ldots b_m]^T$, of order $n + m + 1$. However, alternative multivariable or discrete-time TF models can be accommodated within the same mathematical and software framework (Young and Ratto, 2009, 2011).

[1] See http://captaintoolbox.co.uk/Captain_Toolbox.html and http://www.es.lancs.ac.uk/cres/captain/.

It is clear that the definition of the mapping $\theta(\mathbf{X})$, which is formulated in general, multivariable ($\mathscr{R}^p \rightarrow \mathscr{R}^{(m+n+1)}$) terms, can be quite complex to derive, visualise and evaluate for each element of $\theta(\mathbf{X})$. The emulation modelling procedure considered in the present paper is part of the *Data Based Mechanistic* (DBM) approach to modelling Young and Ratto (2009). In particular, both the identification and estimation of the low order model structure and the mapping $\theta(\mathbf{X})$ are identified using objective criteria, both statistical and interpretative. In order to facilitate the identification and estimation process, however, a clear visualisation of the mapping is required, together with both its uncertainty and/or sensitivity measures.

This paper presents a unified software framework, built within the Matlab software environment, which formalises and implements the identification, estimation and validation stages of the multivariable mapping in a set of easily implemented procedures. These can be modified by the user so that they can be used with various large model types, whether implemented in Matlab, or within a separate application, potentially running on another computer.

The procedures are also easily parallelised, due to the repetitive character of the computations. This can be achieved using either the Matlab Parallel Toolbox and minimally adapted existing code; utilizing the Matlab parfor construct; or explicitly split into a number of compiled Matlab routines run on parallel high performance clusters, such as the Lancaster University High End Computing Cluster (HEC) which was used to run this application. As a feasibility experiment, in the latter regard, the estimation stage used in the later example (which is the slowest step in this example) was run with a different, densely-defined grid on 10 nodes of the Lancaster HEC. The grid was simply divided uniformly into 10 slices and each of the slices was processed separately, in parallel, at the grid nodes. The results were then assembled, producing a complete data structure containing the estimated TF model parameters at the grid nodes, so that further steps of the analysis were unaffected. While this exercise was partly manual, it would be possible to automate it using a combination of Matlab and simple Unix Shell scripts.

One of the important characteristics of the proposed approach is the inherent ease with which the results can be visualised at every stage of the emulation procedure. This is vital, since the user has to assess the mapping results in order to make decisions regarding approximation or parameterization of the generated maps. And this assessment is difficult because operating in parameter spaces of dimension higher than three is quite unintuitive and so difficult to imagine.

3. Multivariable emulation framework tools

Fig. 1 is a block diagram of the DBM emulation procedure. The four stages of the DBM emulation model development shown in this diagram are as follows:

1. Identification and estimation of a nominal TF model: On a nominal set of large model parameters ($\overline{\mathbf{X}} = (\overline{X}_1, \ldots, \overline{X}_p)$), *Dominant Mode Analysis* (DMA: see Young (1999)) is used to provide an identifiable, reduced order TF model that reproduces the response $\mathbf{y}(t)$ of the large model as well as possible. This large model response is obtained using a 'reference' set of forcing input data $\mathbf{u}(t)$, while $\overline{\mathbf{X}}$ is typically set at the mean of the distributions used for the parameters of the large simulation model. This reduced order TF model constitutes the *Nominal Emulation Model* (NEM).
2. MC randomization: *Monte Carlo* (MC) simulation analysis is carried out, by taking a training set $\mathbf{X}^{(i)}$, $i = 1,\ldots,N_r$, of independent, randomly chosen parameters from the large model, and computing the output response $\mathbf{y}(t)^{(i)}$ of the original large order model, always keeping the same set of forcing input data $\mathbf{u}(t)$ used in the first step. Then, for each MC computer experiment $\{\mathbf{u}(t), \mathbf{y}(t)^{(i)}\}$, a dynamic TF model with the same reduced order obtained in step 1. above is estimated, obtaining a MC set $\boldsymbol{\theta}^{(i)}$ of TF coefficients.
3. Mapping: in order to emulate the large simulator at any untried sample $\mathbf{X}^*$, a mapping between the TF models estimated in step 2. and the $\mathbf{X}$ parameters is required. This mapping can be achieved in two different ways: 'stand-alone parameter mapping' and 'response mapping' modes.
4. Validation, uncertainty and sensitivity analysis: the previous three steps yield the *Full Emulation Model* (FEM). This FEM can be validated by generating dynamic simulations $\mathbf{y}(\mathbf{u}^*(t)|\mathbf{X}^*, t)$ at untried $\mathbf{X}^*$, for any new sequence of forcing inputs $\mathbf{u}^*(t)$, and then comparing this with the FEM output response $\hat{\mathbf{y}}(t)$ to the same forcing inputs $\mathbf{u}^*(t)$. A complete statistical uncertainty estimation can be derived for the surrogate dynamic behaviour, as well as a comprehensive sensitivity analysis.

Note that 'response mapping' is a version of emulation approach where, for a given set of inputs and a set (grid) of large model parameters, a set of outputs is generated, and the emulator's response to this set of inputs is obtained by mapping using the stored outputs directly. This approach is not considered in the present paper but further details are available in other published papers Young and Ratto (2011); Higdon et al. (2007); Bayarri et al. (2007).

The NEM is quite simple since it relies mainly on the well tried ability of the CAPTAIN transfer function identification and estimation routines to perform the model reduction that is the core of DMA. As a result, the dynamic behaviour of the nominal emulation model is normally almost identical to that of a large linear computer simulation model and, depending on the nature of nonlinear dynamics, can reproduce well the behaviour of large nonlinear models, as discussed briefly in the conclusions Section 5. On the other hand, the development of the FEM presents a much more complex computational problem because it requires the manipulation of data in multivariable parametric spaces of different, and potentially high, dimensions. And, as Young and Ratto (2011) point out, it includes approximations that can reduce the emulation accuracy when compared with that of the NEM. In order to handle this complex multi-dimensional approximation procedure, the present framework uses three main p-dimensional Matlab structures for any emulation task, where p is the number of mapped parameters of the original large model.

3.1. Sampling considerations and the main data structures

The emulation framework uses a p-dimensional grid defined by the user, where each of the dimensions represents one of the parameters of the large model, and is split into a number of points defining the nodes. In the current version, the grid defines a hypercube in p-dimensions, uniformly divided along each of its dimensions. At this stage we introduce the index I which spans the whole grid, numbering all the nodes. Note how the index $i \subseteq I$ defined above (in stage 2. of the DBM emulation procedure) can form a randomised subset of I. Nodes of the grid indexed by i are where the simulations of the large model are performed and the transfer function model parameters estimated.

The uniformity of this large model parameters grid is dictated by the requirement for flexible visualisation of estimated low order

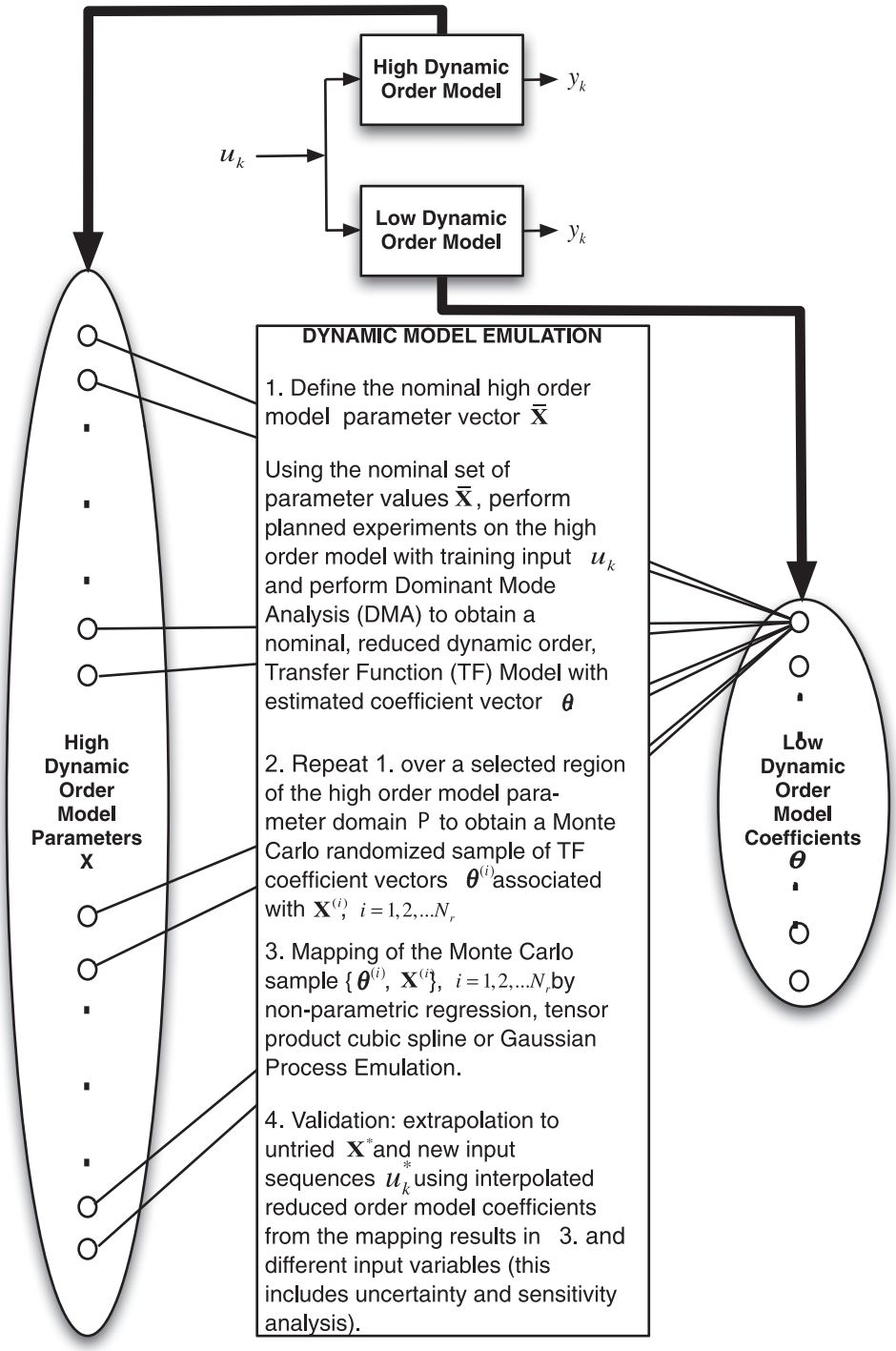

Fig. 1. The process of emulation model synthesis: the items 1–4 in the central block link with those in the text (after Young and Ratto (2011)).

model parameters embedded within the DBM approach. However, the large model parameters can be sampled within the large model parameter space in a variety of ways, and the uniform grid required for visualisation can be 'filled in' after the simulation and estimation stages, during the mapping and DBM identification stage. For example the user could use a (potentially fine) uniform grid, subsampled using any available strategy (e.g. a version of Monte Carlo sampling, as in Step 2 above) and, during Step 3 above, apply an interpolation procedure, such as the Smoothing Spline ANOVA procedure used in Young and Ratto (2011), to fill the remaining grid nodes. In other words, the full uniform grid used in the later example is not a limitation of the proposed data framework: existing sampling and interpolation procedures can be adopted and 'plugged into' the process, reducing significantly the number of simulations of the full model and of the estimation procedure analysis. This is due to the open character of this data framework, made possible by the generality of the data structures and available Matlab coding.

Every stage of the emulation procedure has its associated *p*-dimensional data structure constructed on the same *p*-dimensional

grid of the large model parameters. There are three data structures, which contain the results of the subsequent stages in the emulation process:

(i) The *ParSpace* structure contains the parameters $\mathbf{X}$ of the large model: each node I includes the values $\mathbf{X}^{(I)}$, where I defined above is the set of indices spanning the entire p-dimensional grid-space.
(ii) The *OutSpace* structure contains outputs of the large model $\mathbf{y}(t)^{(i)}$ given its chosen input(s), obtained from simulation of the large model for the selected subset $\mathbf{X}^{(i)}$; $i \subseteq I$ of its parameter sets – the *ParSpace* nodes.
(iii) The *EstSpace* structure of the same dimensionality contains in its nodes indexed by i the low order transfer function models (model objects, potentially multiple), including parameter vectors $\boldsymbol{\theta}^{(i)}$ and their uncertainty information, as well as other quantities contained in the standard Matlab Identification Toolbox *idpoly* objects (see for example http://www.mathworks.com/help/toolbox/ident/ref/idpoly.html), which are also returned by the Captain Toolbox transfer function estimation procedures. These objects are obtained using transfer function estimation procedures with model structure identification carried out by the rivcbjid routine; and subsequent final estimation of the parameters that characterize this identified model structure accomplished by application of the rivcbj routine. The estimation procedures use the input–output data contained in the *OutSpace* structure. Since in general the *OutSpace* structure's grid may be sub-sampled at the nodes $i \subseteq I$, the estimation is carried out at the nodes i where simulations were carried out; subsequent p-dimensional interpolation (or approximation/extrapolation, as appropriate) expanding the mapping to the whole grid I can then be used at the mapping stage and so complete the grid for visualisation and DBM identification purposes.

These structures can be linked with the steps in the DBM emulation modelling procedure outlined above. In particular, stage (1.) relates not only to the sampling of the large model parameter space, but also to the identification and estimation of the low order TF model. There is also an unavoidable preparatory step of experimenting with the feasible TF model structures over the full large model parameter range in order to ensure that the entire parameter sub-space of the model can be adequately mapped into its TF counterpart. Mapping is necessarily limited to the user-specified range of large model parameters that are considered adequate for the required modelling task. Naturally, the selected TF model structure needs to emulate the large model well over the whole of this range.

If it is required, the most time consuming processes of 'filling' the, potentially sub-sampled, *OutSpace* and *EstSpace* structures (large model simulations and estimation of the transfer function models) can be parallelised quite easily, either by manually splitting the *ParSpace* into 'slices' run as separate processes, or through the use of the Matlab parallel tools. The linking of the *ParSpace* and *EstSpace* structures is the inherent part of the mapping step of DBM emulation. The full mapping visualisation requires TF model parameter values being available at all the nodes on the grid as indexed by I. TF parameters will be either estimated from the Input–Output data (over the $i \subseteq I$ subset of the grid), or interpolated (or approximated) over the remainder of the full grid: I i during the mapping process.

The validation as well as uncertainty and sensitivity analysis are facilitated by the existing data structures, as shown in the example below. Each of the plots, including annotations, is produced automatically by a single call to a visualisation function, so that the whole framework would easily lend itself to being absorbed into a full GUI-based package.

3.2. Functions

The data structures largely define the functionality of any Matlab routines using them but, while a basic suite of general functions are provided, the users will naturally need to 'plug in' their simulation software, if these are available in Matlab or Simulink, or access them by calls to the external modules. This is facilitated by the strongly modular structure of the suite of functions and example scripts.

Functions are also provided to extract and combine the mapping information from the *ParSpace* and *EstSpace* structures in a form suitable for use with Matlab's multivariable visualisation tools, which use the same type of grid. Other functions include 2D and 3D projection visualisations with automated graph annotations, as shown in the example below. This multivariable visualisation is used to assess the complexity of the sample mappings and to choose a suitable approximation, whether through an interpolation, parameterisation or a non-parametric approximation. While they are facilitated by the clear data structures, these problem-specific steps need to be implemented by the user.

4. Illustrative example: emulation of the OTIS computer model

This example illustrates how the procedures and software framework discussed above have been applied to the large OTIS computer model which simulates the distributed parameter *Transient Storage* (TS) model originally introduced by Coates (Coats and Smith, 1964) and subsequently used in numerous hydrological modelling applications (e.g., see Valentine and Wood, 1977; Wagener et al., 2002; Martinez and Wise, 2003). It is a modification of the classical *Advection-Dispersion Equation* (ADE) model that has become popular because the ADE, on its own, is unable to explain the asymmetry, in the form of the extended 'tail' of the impulse ('gulp') response curves, that is observed in most real river channels.

It is interesting to note that the TS model helped to stimulate the development of the *Aggregated Dead Zone* (ADZ) model Beer and Young (1983); Wallis et al. (1989). This much simpler, lumped parameter model is described by simple ordinary differential equations or their discrete-time counterparts, which are directly equivalent to the transfer function model used below for reduced order emulation of the OTIS model. Indeed, the poor identifiability of the TS model parameters indicated by Wagener et al Wagener et al. (2002), especially when compared to the very good identifiability of the ADZ model, was another reason for us to investigate the TS model in the context of links between it and a TF emulation model.

4.1. The OTIS model

The TS model is expressed as two coupled equations that describe the transport and dispersion of a solute in a river channel: a standard, partial differential ADE for the concentration $C(t, x)$ of solute in the main channel, linked by an exchange coefficient α to an ordinary differential equation that describes the variation of the solute concentration $C_s(t)$ in the storage zone:

$$\begin{aligned}\frac{\partial C(t,x)}{\partial t} &= -\frac{Q}{A}\frac{\partial C(t,x)}{\partial x} + \frac{1}{A}\frac{\partial}{\partial x}\left(AD\frac{\partial C(t,x)}{\partial x}\right) + \alpha(C_s(t,x) - C(t,x)) \\ \frac{dC_s(t,x)}{dt} &= -\alpha\frac{A}{A_s}(C_s(t,x) - C(t,x)) \end{aligned} \quad (2)$$

Here t denotes time; x is the distance along the main channel; Q is the volumetric flow; A is the main channel cross-sectional area; α is

the exchange coefficient; A_s is the storage zone cross-sectional area; and D is the longitudinal dispersion coefficient.

The OTIS model developed by Runkel et al. (1996) expands the original TS model by including the solute's sorption and release, lateral inflow etc. It has been selected to illustrate the application of the proposed emulation software environment because it is a well known and very popular simulation model that provides a reasonable example for the full emulation of a linear, distributed parameter system. Crucially for the present study, however, the developers of OTIS provide a Fortran simulation module and full documentation, including Fortran source code and several binaries (see http://water.usgs.gov/software/OTIS/). The model is not being used here to demonstrate the efficacy of DBM model emulation although, together with the YR publications, it does illustrate the potential of this approach. In this regard, other publications (e.g. Young et al. (2009)) have considered how the approach can be extended to the emulation of nonlinear systems and this is discussed briefly in the Conclusions Section 5.

4.2. Selection of the parameters and suitable simulations of the OTIS model

The specific OTIS model used in the present example is that developed by Chris Martinez and William R. Wise of the Environmental Engineering Sciences Department, University of Florida Martinez and Wise (2003) in their analysis of data obtained from a tracer experiment in a wetland area using conservative potassium bromide (KBr) as the tracer material. This wetland area is located in Florida, USA, and it receives treated domestic wastewater which travels slowly through the wetland to allow for further nutrient removal. The tracer experiment was part of a study carried out by Martinez and Wise for the City of Orlando. Their study objective was to determine residence times for each wetland cell in the system and to assess whether the same degree of treatment could be maintained should the wastewater loading be raised from 16 to 20 million gallons per day. The model structure and nominal parameters are based on those inferred by Martinez and Wise for Cell 12 in the wetland area. Three of the five main parameters of the full OTIS model have been selected for this example, with no compromise to the generality of the software: the Dispersion coefficient D, the contact area of the transient storage zone A_s and the additional scaling coefficient α. The nominal values of these parameters for Cell 12 are: $D = 0.130m^2/s$, $A_s = 72.1m^2$ and $\alpha = 1.27\times10^{-6}s^{-1}$. The remaining nominal parameters in Eq. (2) are set to the values specified for Cell 12 by Martinez and Wise: $Q = 0.130m^3/s$, $A = 67.8m^2$, and the section length $L = 755m$.

It is worth pointing out here that, for constant Q and A, an analytical solution of the TS equations has been developed by de Smedt (2006) for specified inputs. The same study shows how the OTIS simulations are consistent with the analytical solutions, where the latter exist. However, the OTIS model continues to be well used in practice because of its flexibility and, so far, such analytical solutions have not been popular in the hydrological literature. Consequently, the emulation of the OTIS model has considerable practical relevance. Indeed, this popularity is one reason for using the model as an illustrative example.

It is important to ensure that the emulator transfer function model can be successfully estimated for all the points in the user-specified parameter space of the full model and for a wide range of input perturbations. To some degree, this can restrict the range of variation of the full model parameters that can be handled in a single emulation exercise. However, it is not a limitation in general terms, since other sub-spaces can be defined with other structures of the transfer function model. The range of variation assumed for each of the parameters is currently specified by the user as relative variation from the nominal parameter values in $\overline{\mathbf{X}} = [0.130\ 72.1\ 1.27\times10^{-6}]^T$ and, in this case, we applied multiplicative variations of (−0.3,0.3), (−0.5,2), (−0.9,1.5), respectively: for instance, this means that the nominal value of $A_s = 72.1$ is varied from between a half and double its nominal value. The ranges are selected based on the physical interpretation of the large model parameters; the requirement to cover a broad range of the large model behaviours; and the need to satisfactorily identify and estimate a TF model at all the nodes. This latter requirement was not much of a constraint in this exercise because the rivcbjid and rivcbj routines in CAPTAIN that were used for this purpose are so robust. The relatively narrow variation of the D coefficient was dictated by its significant influence on the observed dynamics, as shown in Fig. 2, which shows a small selection of the impulse responses used in the estimation of the TF models, as discussed in the next sub-section.

The *ParSpace* structure was used here to generate OTIS control files and run the executable outside Matlab for all the nodes (there was no subsampling in this simple example). Each run of OTIS produced its output files, which were then read into the nodes of the *OutSpace* structure.

4.3. Estimation and parametric mapping results

Following previous experiments with similar types of PDE model Young (2011), the OTIS model was perturbed by both impulse and repeated step inputs. For all the responses generated with such inputs, a 6th order, continuous-time transfer function model structure, [6, 6τ], was identified for this emulation experiment using the rivcbjid routine in the CAPTAIN Toolbox, with τ chosen to allow for the changing advective time delays in the large model as its parameters are varied. In all cases, the response data were explained almost perfectly, with the simulation coefficient of determination (Nash-Sutcliffe coefficient of efficiency), R_T^2, based on the error between the OTIS model response and the emulation model output, always greater than 0.999 (i.e. >99.9% of the OTIS model output variance explained by the emulation model). Conveniently, these identified 6th order TF models were identifiable throughout the sampled hyper-cube of the parameters of the OTIS model, as required for stand-alone emulation without the introduction of sub-spaces into the parametric maps (see earlier

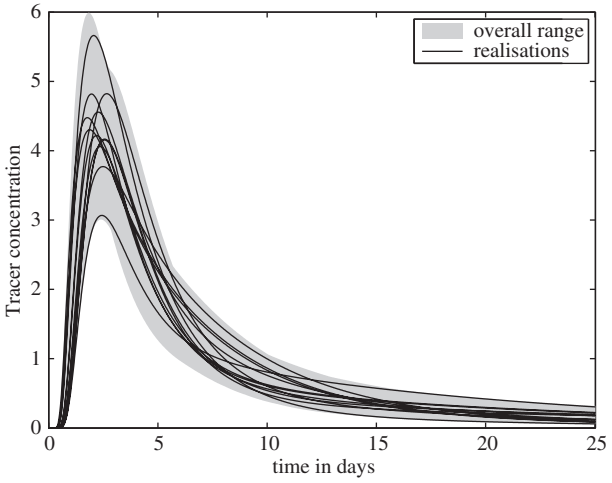

Fig. 2. The set of OTIS simulation outputs: the range of responses (concentration in mg/l) to impulse ('gulp') injection of tracer solute simulated over the full grid of the OTIS model parameters. For clarity, the overall range is shown as grey outline, with 12 randomly chosen realisations included.

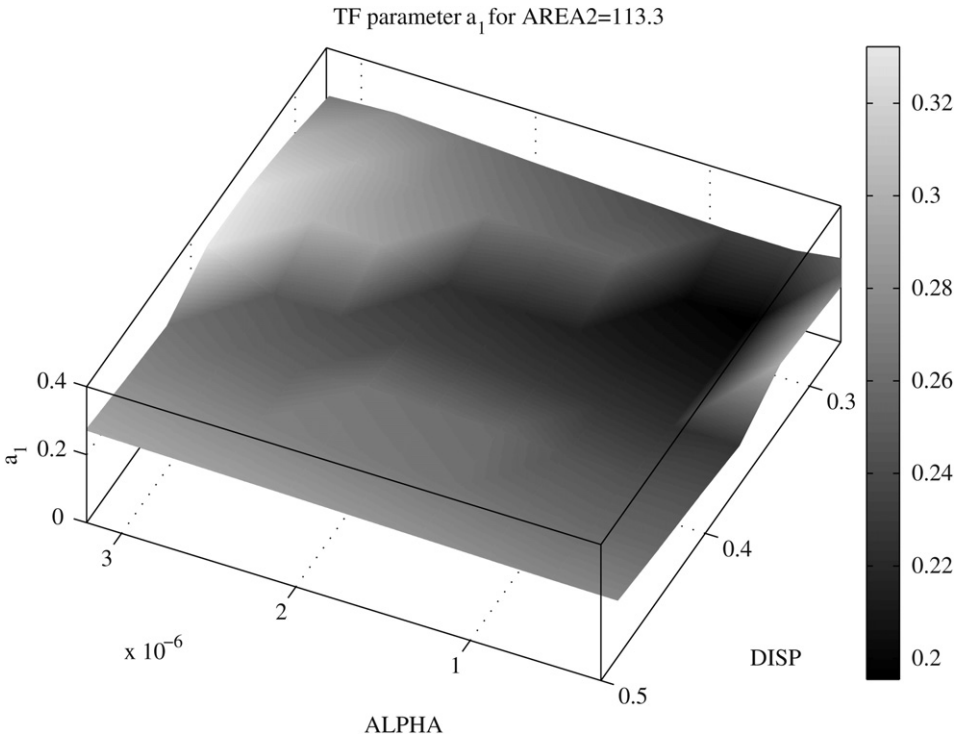

Fig. 3. 3D projection of the mapping of the a_1 parameter of the emulating transfer function from α (termed ALPHA in the OTIS parameters list) and D (termed DISP in OTIS) parameters of the full OTIS model, with A_s (AREA2 in OTIS) fixed. Note that in this and subsequent automatically generated figures the original names of OTIS parameters are used, and clarified in the captions. All the plots generated by the software are labelled automatically, using the OTIS configuration file nomenclature.

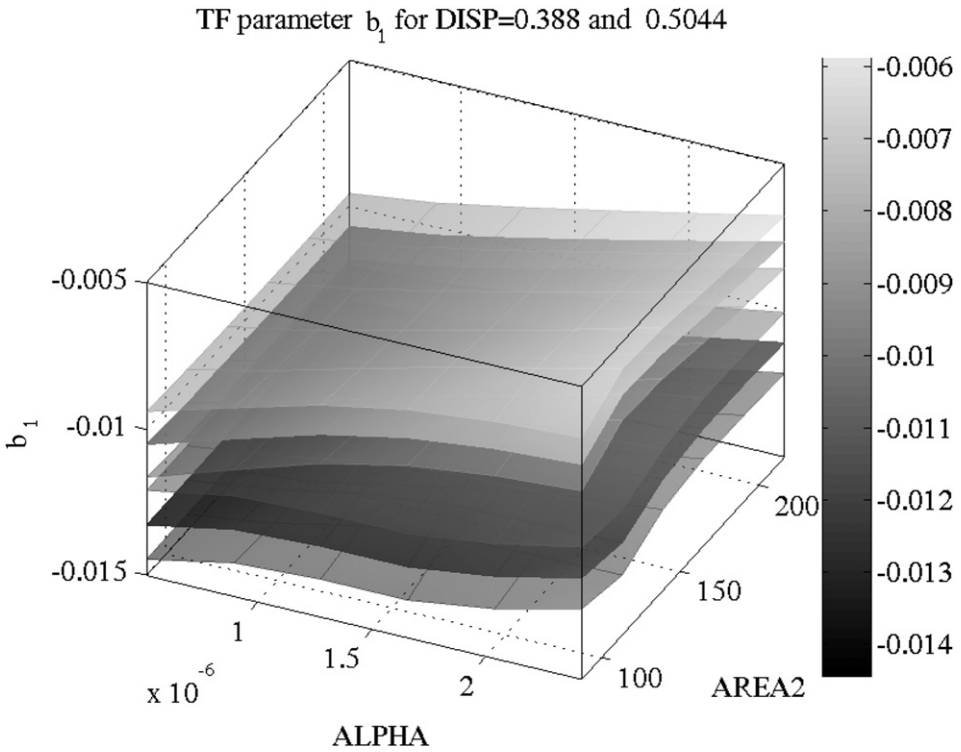

Fig. 4. Example of 3D projections of the mapping of the b_1 parameter of the transfer function. The projections are shown with two solid surfaces representing two different values of the diffusion coefficient D (DISP in OTIS) of 0.349 and 0.5044, with their associated estimation uncertainties (semi-transparent surfaces). The visualisation is 'zoomed' into a smaller range of the (α, A_s) plane – also a pre-programmed feature of the package.

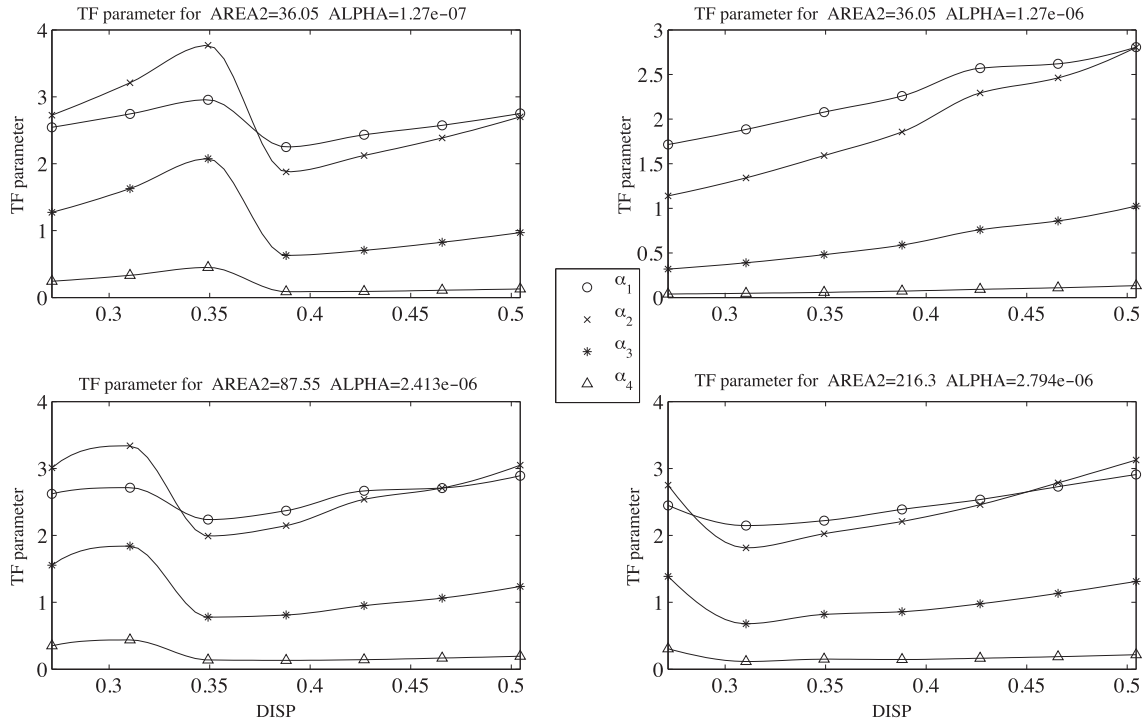

Fig. 5. 2D 'slices' of the same relationship as shown in Fig. 3 for the first four *a* parameters (denominator coefficients of the transfer function) shown as functions of the diffusion coefficient *D* (termed DIFF in OTIS) with two large model parameters frozen at selected values shown in the figure. OTIS AREA2 and ALPHA parameters (A_s and α) are fixed at values shown in the plots.

comment). Note that, although there are 13 parameters in the [6, 6τ] TF model and only five main parameters in the OTIS model, the TF model is much more computationally efficient, with an average computational time ratio of 940 (i.e. the TF model simulation time is 940 times faster than that of the OTIS model). In the more general situation, of course, the large model can have many more parameters than the nominal emulation, as in the case of the global carbon cycle models emulated by Young et al. (1996a); Young (1998); Young and Parkinson (2002); and the large power generation simulation model considered by Lu et al. (2011).

A reviewer pointed out that the identification of a 6th order model in this example appeared to contrast with the results obtained by Lees et al. (2000) (see earlier comments in Section 1), where a first order ADZ model is apparently used to emulate the response of the OTIS model. However, the Lees et al results are rather misleading when interpreted in full emulation terms. First, each such ADZ model is emulating the behaviour of only a single reach, so that a six reach model would require a 6th order TF composed of the six ADZ models in series (similar to the 6th order TF identified here). Second, the inputs used by them are limited to the relatively 'smooth' inputs associated with tracer experiments. While these allow for the identification of lower order models that are useful for analyzing tracer data, they are not sufficient for full model emulation, which requires a wider range of input perturbations. In full model emulation, it is necessary to fit to the responses generated by inputs that are consistent with the requirement for 'sufficient excitation' when planning dynamic experiments of this type, such as the impulse response and repeated step inputs utilized in this example. Finally, note that even in the special situation considered by Lees et al, there are clear differences between the ADZ and TS outputs in their Fig. 6: their explanation of the data is not sufficient for full emulation modelling situation, where coefficients of determination R_T^2, based on the simulation modelling error, normally need to be greater than 0.999, as achieved in the present analysis.

Examples of the emulation mapping results (for coarse subdivisions in the model parameter space) are presented in Figs. 3–5. Fig. 3 shows the mapping of α and *D* into the first parameter of the transfer function denominator polynomial, with the A_s value fixed near its mid-range. It is worth noting that the figure is generated automatically by the emulation framework software, including its captions, which facilitates the study of the emulation results. The resolution of about 20 grid points along each of the axes of the large model parameter space (3 in this example) is shown in the plot.

A different projection is presented in Fig. 4, showing the mapping of β_1 based on α and A_s, with *D* fixed at two values (two slices). It also reveals the 95% confidence intervals of the estimated transfer function parameter using semi-transparent surfaces and is an example of how the visualisation routines have a built-in zoom function that allows the user to inspect local regions of the

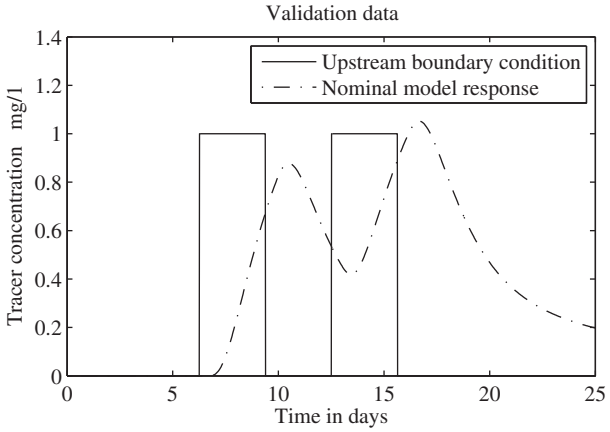

Fig. 6. The independent input (full line – scaled) and large model response (concentration in mg/l) used in the validation of the emulator.

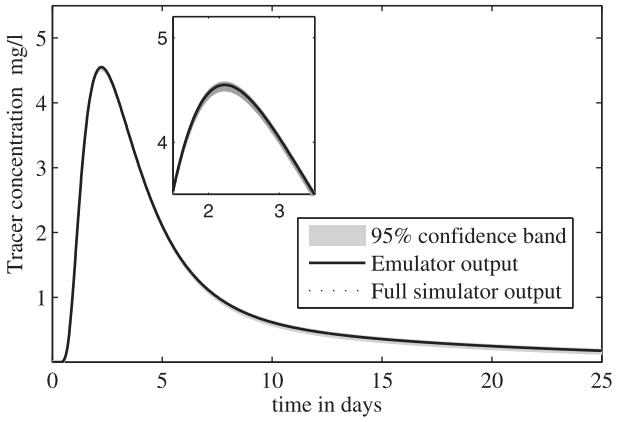

Fig. 7. OTIS and emulation model outputs (concentration in mg/l) for a single impulse input and a randomly selected point in the full model parameter space. The basic grid used throughout the example is used here (about 8 divisions in each large model parameter range, as visible in the mapping of Fig. 3).

multivariable plots. In this example, linear interpolation is used to obtain the TF parameter values between nodes. In this regard, note that, in this uniform grid version, we expect that, at the mapping stage, users will utilize either SS ANOVA (as in YR) or *Piecewise Cubic Hermite Interpolating Polynomial* (Fritsch and Carlson, 1980; Kahaner and Nash, 1988) (PCHIP) splines, with the latter preserving the values at the nodes and montonicity in the interpolation. However, as noted above, the data structures that are provided in the software do not limit the users in the choice of the higher dimensional interpolation/approximation prodedures. Other approaches, such as kriging, potential functions, non-parametric regression, tensor product cubic spline smoothing or Gaussian process emulation can be 'plugged-in' easily into the available data 'sockets' to replace the simple procedures used in this illustrative example.

Fig. 5 shows 2D slices (projections onto D parameter axis) of the first three denominator parameter maps, with four different pairs of fixed A_s and α as functions of D. The individual figures, with captions, are generated automatically, on demand, when the user specifies variables fixed at selected values. Only the first four denominator parameters are shown in this plot because the remaining parameters are too small at this scale and so would not provide any additional information. PCHIP spline interpolation is used in this figure for illustration only, in order to show the desirable properties of PCHIP splines in this context. When multivariable PCHIP splines become available in Matlab, this will most likely become the default interpolation method. As in Fig. 4, it is easy to add the uncertainty information to such 2D plots, which facilitates any visual assessment of the mapping complexity.

The above visualisations considerably facilitate the process of identifying the subsequent stages of the emulator development procedure, particularly the choice of the approximation or interpolation methods associated with the mapping between the active nodes. Factors, such as the complexity of the resulting hypersurfaces and the existence of possible non-differentiable areas, need to be detected and assessed before any decisions are made regarding the methods used in the further analysis, which are discussed in the next sub-section. Effective visualisation will also help to identify parts of the large model parameter space where more dense sampling should be applied.

4.4. Stand-alone parametric emulation and validation

The validation stage is intended to ensure that the emulator reproduces well the response of the large model at randomly selected points on the hyper-dimensional map, both for the inputs used in the synthesis of the map and for other, independent inputs. In particular, for each validation input, it involves running both the reduced order TF model and the full OTIS solver over a user-selected number (here 1000 are specified) randomly sampled points in the p-dimensional (here $p = 3$) full model parameter space. All of these sampled locations lie within the interpolated regions so that the validation tests out the approximation introduced by the interpolation process and provides a very good assessment of the emulation performance.

In this example, the independent input signal was the 'pollutant release' scenario plotted as the dash-dot line in Fig. 6, which shows this input and the resultant response of the large model with its parameters set to their nominal values (the reduced order TF emulation model response is virtually identical and so this is not

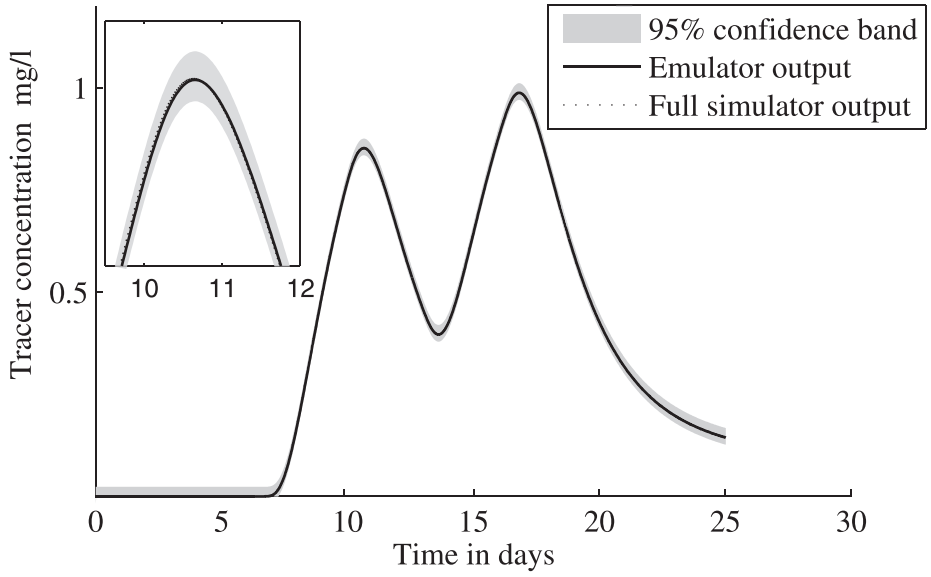

Fig. 8. A typical validation result showing the OTIS and emulation model outputs (concentration in mg/l) for a randomly selected point in the full model parameter space using the validation input signal shown in Fig. 6. As in Fig. 7, the shaded area is the 95% uncertainty band and the full model simulation line is practically indistinguishable from the emulator output.

plotted). This input is a good validation test since it involves harsh step changes that provide a demanding test for the precision of the numerical solution of both the OTIS and TF models. In this context, it is interesting to note that OTIS, in the default executable form used here, does not allow for arbitrary input signals because the upstream boundary value definitions only permit 200 changes of their values. Whilst this is suitable for most tracer experiments, the user would have to recompile OTIS sources to allow for the use of continuous inputs. Of course the TF-based emulator has no such restrictions, showing how stand-alone emulators, such as this, are capable of working around some limitations of the original large models that arise because of their high numerical complexity.

Figs. 7 and 8 show the results of the two validation exercises. Here, one thousand, three dimensional MC realisations of the three selected large model parameters were selected from a uniform distribution in three dimensions and used to run both the OTIS model and the emulator. The simulation errors were recorded and their empirical distribution calculated, as shown in the figures. As can be seen, the difference between the large model and emulator responses is very small at this scale (see insets).

Simple linear interpolation of the parametric mapping in the p dimensions proved to be perfectly satisfactory in this example and generated the good results shown in Figs. 7 and 8. Although the spline interpolation is useful in visualisation terms, it does not give equally good results in this situation, probably because cubic splines can produce non-monotonic artefacts. Other promising interpolation approaches include the PCHIP spline mentioned previously. This is an approximation that preserves monotonicity and is available in both the standard Matlab library and Octave (the GNU Matlab equivalent). However, it is only implemented in Matlab for 1-dimension, and in Octave for 2-dimensions, so it has not been applied here, although it will be subject to future development because of its useful properties.

5. Conclusions

This paper describes a set of software tools that not only service the basic computational requirements of the DBM approach to dynamic emulation modelling for linear systems, as suggested by (Young and Ratto, 2009, 2011), but also enhances the methodology by facilitating the human-interactive element that is an essential aspect of this approach. The resulting software routines provide a unified platform for modellers wishing to use and develop this promising technique and we hope that, by supplying such a set of quite general but advanced routines, we may encourage joint work and open development of this emerging technology.

Based on our experience with interpolation and smoothing between the nodes, as well as with grids of varying resolution (from 4 to 20 divisions along each of the large model parameter dimensions), we feel that the dependence of the emulation results on the grid or simulation nodes in these experiments will normally have to be resolved on a case-by-case basis. However, a set of effective and general software tools that facilitates the evaluation of the mapping, such as those described in this paper, should help to improve the emulation modelling efficiency and make the approach more systematic.

The DBM model emulation used in the illustrative example is limited in two regards: it is applied to the emulation of a large linear simulation model (albeit a distributed parameter model solved numerically) and there is only a single input perturbation. However, the DBM modelling approach to emulation is not limited in these ways. As regards nonlinear systems, sometimes linear models can be very effective in modelling nonlinear dynamics, even for quite large response perturbations: see e.g. Young et al. (1996b); Young (1998); Young and Parkinson (2002), where linear models are used for the nominal emulation of nonlinear global carbon cycle models, as used in climate change studies; and Lu et al. (2011), who estimate linear, multi-input models for the nominal emulation of large nonlinear power generation simulation model over a range of different operating conditions and then use the resulting multivariable emulation model in control system design studies.

If the nonlinear dynamics are more complex and it becomes essential to include nonlinear dynamic elements in the emulation model, the most obvious nonlinear extension of the TF emulation models is to identify the location and nature of state-dependent parameters in the TF using *State-Dependent Parameter* (SDP) estimation: see e.g. Young et al. (2001); Young (2011); Beven et al. (2011). A recent example of this approach has been applied to the emulation of a 89 cross-sectional node HEC-RAS simulation model of the River Severn in the UK Young et al. (2009). Such extensions are easily accommodated into the proposed software environment because they just involve the addition of parameters associated with the parameterization of the SDP nonlinearities.

Although the TF estimation methods available in the CAPTAIN Toolbox have been implemented in a full multivariable form Jakeman and Young (1979), the resulting algorithms are not very easy to use in practice. Consequently, the resulting practical restriction to single or multi-input systems limits the approach to multivariable systems that can be modelled adequately in this manner. Whether this is an important limitation will depend on the nature of the large multivariable simulation model. For example, it presented no problems at all in the case of the large, nonlinear power generation simulation model mentioned above, where third order, multiple input, single output sub-models were able to emulate the full multivariable, nonlinear model very well indeed. However, it is clear that, although the DBM modelling approach to emulation has wide application potential, more research is required to establish the limitations of the approach and to define more clearly the class of multivariable, nonlinear systems that can be handled in this manner.

The computational time required in the preliminary stages of the emulation procedure depends very much on the number of active grid points (nodes) $i \subseteq I$ in the large model parameter space. For each of these nodes one simulation of the large model is required, and this is followed by the estimation of a TF model with the identified model structure. In the illustrative example, an estimation run of this type takes a similar time to that required for running the large OTIS simulation model executable, including its call from Matlab (both taking about 1 s on a relatively current PC); and this becomes longer if a range of time delays is searched. This can be compared with a few milliseconds required for an emulation run – three orders of magnitude faster for a relatively fast TS model. Of course, for truly large and complex models, the model run-time will be the critical timing element and this speedup ratio will be much higher. In relation to the time required for emulation model estimation, the interpolation of the transfer function model parameters and the execution of the emulating transfer function model are nearly instantaneous. Naturally, if validation using the parameter mappings does not satisfy the user-specified requirements, then it may be necessary to iterate the analytical procedure until the full emulation model is satisfactory in this regard.

Finally, it is interesting that the emulation study described in this paper, using reduced order continuous-time transfer function models, produces such well behaved solutions for the TS model (in its very general OTIS implementation). This suggests that such a 'stand-alone' emulation model, extended to cover a wider number and range of the OTIS model parameters, could be used as a computationally efficient replacement for the OTIS model. It also raises questions of identifiability, since Wagener et al. (2002) have found that the OTIS model is poorly identifiable from observational

data. In this connection, Young (in press) has found recently that an identifiable, semi-distributed, continuous-time version of the ADZ model might be an alternative to the TS model and can be emulated without difficulty. It has the advantage that it can produce similar dynamic behaviour to that of the TS/OTIS models, without some of their disadvantages, such as longer computation times and the requirement for implicit, finite difference solutions.

Acknowledgements

The authors are grateful to Professor Thorsten Wagener and Dr. Christa Ann Kelleher, Pennsylvania State University, for their advice and for facilitating access to the OTIS computer model; to Chris Martinez and William R. Wise, University of Florida, for supplying the data from their tracer experiment on the Florida wetland; and to the anonymous reviewers whose suggestions considerably improved the paper.

References

Bayarri, M.J., Berger, J.O., Paulo, R., Sacks, J., Cafeo, J.A., Cavendish, J., Lin, C.-H., Tu, J., 2007. A framework for validation of computer models. Technometrics 49.

Beer, T., Young, P.C., 1983. Longitudinal Dispersion in Natural Streams. Journal of Environmental Engineering 109, 1049–1067.

Beven, K.J., Leedal, D.T., Smith, P.J., Young, P.C., in press. Identification and representation of state dependent non-linearities in flood forecasting using the DBM methodology. In: Wang, L., Garnier, H. (Eds.), System Identification, Environmetric Modelling and Control. Springer-Verlag, Berlin.

Coats, K.H., Smith, B.D., 1964. Dead-end pore volume and dispersion in porous media. Soc. Petr. Eng. J. 4, 73–84.

de Smedt, F., 2006. Analytical solutions for transport of decaying solutes in rivers with transient storage. Journal of Hydrology 330, 672–680.

Fritsch, F.N., Carlson, R.E., 1980. Monotone piecewise cubic interpolation. SIAM Journal on Numerical Analysis 17, 238–246.

Higdon, D., Gattiker, J., Williams, B., Rightley, M., 2007. Computer model validation using high-dimensional outputs. In: Bernardo, J., Bayarri, M.J., Berger, J.O., Dawid, A.P., Heckerman, D., Smith, A.F.M., West, M. (Eds.), Bayesian Statistics 8. Oxford Univ. Press.

Jakeman, A.J., Young, P.C., 1979. Refined instrumental variable methods of time-series analysis: part II, multivariable systems. International Journal of Control 29, 621–644.

Kahaner, D.M.C., Nash, S., 1988. Numerical Methods and Software. Prentice Hall.

Lees, M.J., Camacho, L.A., Chapra, S., 2000. On the relationship of transient storage and aggregated dead zone models of longitudinal solute transport in streams. Water Resources Research 36 (1), 213–224.

Liaw, C.M., Pan, C.T., Ouyang, M., 1986. Model reduction of discrete systems using the power decomposition method and the system identification method. IEE Proceedings D, Control Theory and Applications 133 (1), 30–34.

Lu, C.X., Young, P.C., Rees, N.W., 2011. Simulation Model Emulation in Control System Design. In: Wang, L., Garnier, H., Jakeman, A.J. (Eds.), System Identification, Environmetric Modelling and Control. Springer-Verlag, Berlin.

Martinez, C.J., Wise, W.R., 2003. Analysis of constructed treatment wetland hydraulics with the transient storage model OTIS. Ecological Engineering 20, 211–222.

Moore, B., 1981. Principal component analysis in linear systems: Controllability, observability, and model reduction. IEEE Transactions on Automatic Control AC 26 (1), 17–31.

Pedregal, D.J., Taylor, C.J., Young, P.C., 2005. The CAPTAIN Handbook. Centre for Research on Environmental Systems and Statistics. Lancaster University.

Runkel, R.L., Bencala, K.E., Broshears, R.E., Chapra, S.C., 1996. Reactive solute transport in small streams: I. development of an equilibrium-based simulation model. Water Resources Research 32 (2), 409–418.

Santner, T.J., Williams, B., Notz, W., 2003. The Design and Analysis of Computer Experiments. Springer-Verlag.

Taylor, C.J., Pedregal, D.J., Young, P.C., Tych, W., 2007. Environmental time series analysis and forecasting with the CAPTAIN toolbox. Environmental Modelling & Software 22 (6), 797–814.

Valentine, E.M., Wood, I.R., 1977. Longitudinal dispersion with dead zones. Journal of the Hydraulics Division, ASCE 103 (9), 975–990.

Wagener, T., Camacho, L.A., Wheater, H.S., 2002. Dynamic identifiability analysis of the transient storage model for solute transport in rivers. Journal of Hydroinformatics 4, 199–211.

Wallis, S.G., Young, P.C., Beven, K.J., 1989. Experimental investigation of the aggregated dead zone model for longitudinal solute transport in stream channels. Proceeding Institution of Civil Engineers 87 (Part 2), 1–22.

Willcox, K., Peraire, J., 2002. Balanced model reduction via the proper orthogonal decomposition. American Institute of Aeronautics and Astronautics (AIAA) Journal 40 (11), 2323–2330.

Young, P., 1998. Data-based mechanistic modelling of environmental, ecological, economic and engineering systems. Environmental Modelling & Software 13 (2), 105–122.

Young, P., Parkinson, S., Lees, M., 1996a. Simplicity out of complexity in environmental modelling: occam's razor revisited. Journal of Applied Statistics 23 (2), 165–210.

Young, P.C., 1999. Data-based mechanistic modelling, generalised sensitivity and dominant mode analysis. Computer Physics Communications 117, 113–129.

Young, P.C., 2006. The CAPTAIN toolbox for Matlab. In: Proceedings 14th IFAC Symposium on System Identification SYSID06, Newcastle, NSW, Australia. Newcastle, NSW, pp. 909–914.

Young, P.C., 2011. Recursive Estimation and Time-Series Analysis. Springer-Verlag, Berlin (much revised and extended version of the similarly named 1984 book).

Young, P.C., 2011. Data-based Mechanistic Modelling: Natural philosophy revisited? In: Wang, L., Garnier, H., Jakeman, A.J. (Eds.), System Identification, Environmetric Modelling and Control. Springer-Verlag, Berlin.

Young, P.C., Garnier, H., 2006. Identification and estimation of continuous-time, data-based mechanistic models for environmental systems. Environmental Modelling & Software 21, 1055–1072.

Young, P.C., Leedal, D., Beven, K.J., Szczypta, C., 2009. Reduced order emulation of distributed hydraulic models. In: Proceedings 15th IFAC Symposium on system identification SYSID09, St Malo, France.

Young, P.C., McKenna, P., Bruun, J., 2001. Identification of nonlinear stochastic systems by state dependent parameter estimation. International Journal of Control 74, 1837–1857.

Young, P.C., Parkinson, S., 2002. Simplicity out of complexity. In: Beck, M.B. (Ed.), Environmental Foresight and Models: a Manifesto. Elsevier, Oxford, pp. 251–294.

Young, P.C., Parkinson, S., Lees, M.J., 1996b. Simplicity out of complexity: occam's razor revisited. Journal of Applied Statistics 23, 165–210.

Young, P.C., Ratto, M., 2009. A unified approach to environmental systems modeling. Stochastic Environmental Research and Risk Assessment 23, 1037–1057.

Young, P.C., Ratto, M., 2011. Statistical emulation of large linear dynamic models. Technometrics 53, 29–43.