Internal Report
Modular Dynamic Emulation and Internal Model Discrepancy for a Rainfall Runoff Model.

Ian Vernon, Allan Seheult & Michael Goldstein

Department of Mathematical Sciences, Durham University,
Science Laboratories, Stockton Road, Durham DH1 3LE, UK

October 29, 2010

Abstract

We construct a simple modular dynamic emulator for a rainfall runoff model. We investigate the accuracy of such an emulator and discuss the difficulties of such an approach. The emulator is used to investigate the effects of certain internal uncertainties on the outputs of the model, which would be integral to a full reification analysis.

Keywords: computer models, emulation, dynamic emulation, modular emulation, rainfall runoff model, internal model discrepancy, reification.

1 Introduction

Mathematical models of complex physical systems, such as those for climate and floods are usually implemented as a computer code, referred to as a "simulator". When a simulator is fast to run, we can compare it directly to field data. When a simulator is slow to run, we construct a statistical approximation known as an emulator; see, for example, MUCM (2009).

We consider a particular deterministic rainfall runoff model to illustrate the methodology we develop here. The model, which is a discrete time version of a system of differential equations, simulates fluctuations in water discharge and Calcium and Silica concentrations over time. While this model is fast to simulate, we choose to emulate it dynamically; that is, we build a single time-step emulator and run this emulator through time to emulate the complete simulator output. Dynamic emulation takes advantage of our knowledge of the internal simulator processes; for example, for the rainfall runoff model we only need to know the state of the model, the model inputs and the forcing functions at any time t to simulate its state at time $t + 1$. In this account, we go further and emulate three internal modules of the single time-step simulator, which we refer to as "modular dynamic emulation". Here we need to know the state vector of each module, the inputs and outputs to each module, as well the model inputs and forcing functions at any time t in order to emulate the process at time $t + 1$.

A key additional general issue we consider in this report is the inevitable mismatch between a mathematical model and the real system it purports to model. Our approach to account for this discrepancy, employs "reification", a concept developed in Goldstein and Rougier (2009). The idea is to formulate beliefs about a super model, called the "reified model", which incorporates all currently envisaged improvements to the current model.

While we can't build the reified model, at least not right now, we can perhaps formulate an emulator of it. The reified model is considered to be 'closer' to the real system than the current model. Uncertainties in the emulator of the reified model feed through to increase the uncertainties in predicting the real system. Thus, reification is a process whereby the variance of the discrepancy between the current model and the real system is realistically and systematically assessed, and therefore not unduly over-optimistic. In this account, we develop modular dynamic emulation methods which would form a key part of such a reification process, allowing for the incorporation of detailed improvements to the model at the modular level.

This report is structured as follows. In Section 2 we summarise the reification process and discuss emulation. In Section 3 we give a detailed account of the rainfall-runoff model. In Section 4 we discuss dynamic and modular emulation and illustrate the modular emulator performance as applied to the runoff model. In Section 5 we make an initial investigation into the propagation of internal model discrepancies to final output, and in Section 6 we consider future investigations.

2 Reification Strategies

Our eventual goal is to develop a reification strategy for the rainfall runoff model. Essentially three different reification strategies may be considered for the class of such models:

(a) Direct emulation and reification; see, for example, House (2009).

(b) Dynamic emulation and reification.

(c) Modular emulation and reification.

The further we go down this list, the easier reification becomes, whereas the overall system becomes harder to emulate, even though the component modules are themselves often straightforward to emulate. Here we will focus on exploring simple reification ideas using modular dynamic emulators.

2.1 Summary of the Reification Approach

Reification involves linking three models and the system outcomes y, using emulators of each model, the current model f, an improved version f' of f and a reified model f^* that contains all possible improvements to f and f'.

We emulate the current model $f(x)$ using the following standard emulator form for each component of f

$$f_i(x) = \sum_j \beta_{ij}\, g_{ij}(x) + u_i(x) \tag{1}$$

We then ask the subject-matter expert to consider an improved version f' of the model and adjust the emulator accordingly; for example

$$f'_i(x, v) = f_i(x) + \sum_k \gamma_{ik}\, g_{ik}(x, v) + u_i(x, v) \tag{2}$$

where v is a collection of additional inputs.

Then we ask the expert to consider a model f^* that contains all possible improvements to the model and construct the corresponding reified emulator

$$f^*_i(x, v, w) = \sum_j \beta^*_{ij}\, g_{ij}(x) + \sum_k \gamma^*_{ik}\, g_{ik}(x, v) + u^*_i(x) + u^*_i(x, v) + u^*_i(x, v, w) \tag{3}$$

where w is a further additional collection of inputs.

Applying the "best input" approach we link the reified emulator $f_i^*(x, v, w)$ to the real system y by

$$y = f^*(x^*, v^*, w^*) + \epsilon^* \tag{4}$$

where (x^*, v^*, w^*) represents the best input in the reified model. In most other applications, the best input approach is applied to the current model, rather than the reified model.

Reification is often seen to be a difficult and laborious task. In this report we will explore some simple strategies that would form part of a structural reification analysis, House (2009).

2.2 General Emulation Strategy

In order to carry out analysis of a function $f(x)$, we must make many evaluations of the model within a reasonable length of time. For many problems, this is not a realistic possibility. In such cases, we may employ the method of *model emulation*; see, Craig et al. (1997), Craig et al. (2001), Rougier (2009), Kennedy and O'Hagan (2001), O'Hagan (2006), MUCM (2009). Emulation refers to the expression of our beliefs about the function $f(x)$ by means of a fast stochastic representation, which we can use both to approximate the value of the function over the input space and also to assess the uncertainty that we have introduced from using this approximation. For example, we might represent our beliefs about the i-th component of $f(x)$ in the form

$$f_i(x) = \sum_j g_j(x)\, \beta_{ij} + u_i(x) \tag{5}$$

where each $g_j(x)$ is a known deterministic function of x, for example a polynomial term in some sub-collection of the elements of x, the β_{ij} are unknown constants to estimate and $u_i(x)$, the residual function, is specified as having zero mean and constant variance σ_i^2 for each x, with a correlation function $c_i(x, x') = \mathrm{corr}(u_i(x), u_i(x'))$ which only depends on the distance between x and x'. There are many possible choices for the form of the $c_i(x, x')$. If we want to carry out a full probabilistic analysis, then we may suppose, for example, that $u_i(x)$ is a Gaussian process, so that the joint distribution of any sub-collection of values of $u_i(x)$ for different choices of x is multivariate normal.

There is an extensive literature on the construction of emulators for computer models, based on a collection of model evaluations.; see, for example, O'Hagan (2006) and MUCM (2009). Given these evaluations, we may choose our functional forms $g_j(x)$ and estimate the coefficients β_{ij} using standard model building techniques from multiple regression, and then assess the parameters of the residual process $u(x)$ using, for example, variogram methods on the estimated residuals from the fitted model. Given the emulator, we can then carry out any required analysis, but, instead of evaluating the function at each input choice, we evaluate the emulator expectation $\mathrm{E}[f_i(x)]$ at each chosen x. We therefore need to incorporate the emulator variance $\mathrm{Var}[f_i(x)]$ into the analysis in an appropriate manner.

In this report we apply the above general emulation strategy to internal components of the single time-step model, using uncorrelated residual functions. We then make a brief analysis of the effects of various internal uncertainties on certain model outputs.

3 The Rainfall Runoff Model

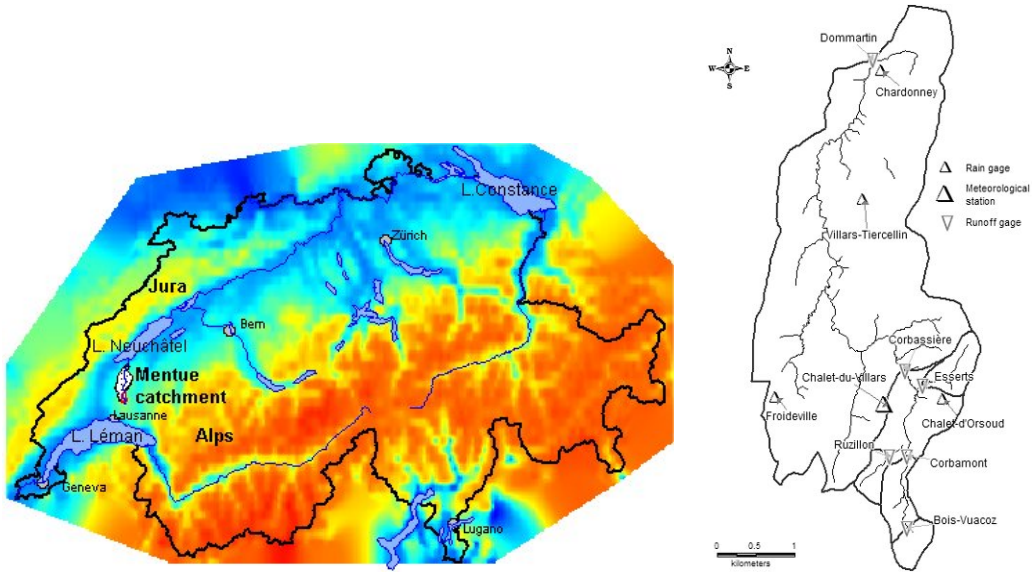


Figure 1: West part of Switzerland, 15 km north of Lausanne, in the canton of Vaud.

We consider a rainfall runoff model described in Iorgulescu et al. (2005) (henceforth IBM), that simulates fluctuations in water discharge and Calcium and Silica concentrations over time. We illustrate our methods with its application to a particular sub-catchment of the Haute-Mentue research catchment (Switzerland); see IBM who refer to other studies and runoff models. Each model run simulates three time series: discharge (D) and the tracers Calcium (Ca) and Silica (Si) over 839 consecutive hours. Any such simulation may be compared to the corresponding 839 hours of field data collected at the sub-catchment between August and September 1993. The field data also includes hourly rainfall which is used as a forcing function (RAIN) to the model. There is a second forcing function, actual evapotranspiration (AET), an evaporation rate, which is modelled as a deterministic sinusoidal function of time.

3.1 The Mathematical Model

The model, depicted in Fig. 2, comprises three compartments with parallel transfer, whereby water, input as rain, may enter three compartments representing three different soil types, "Direct Precipitation" (DP), "Acid Soil" (AS) and "Ground Water" (GW). The water is stored in each compartment for a fast or slow amount of time before being discharged into the streams. The water can instead enter the "Ineffective Storage" compartment, in which case it will not be discharged and can only leave the system via actual evapotranspiration (AET). Six parameters $a_{soil}, b_{soil}, k_{soil}, p_{soil}, c^f_{soil}$ and c^s_{soil} characterise the fluid dynamics of water flow through each *soil* (DP, AS, GW), subject to the constraint $k_{DP} + k_{AS} + k_{GW} = 1$, leaving 17 functionally independent input parameters. Details of parameter descriptions, ranges and units are given in IBM.

Thus, in terms of the general description given in Section 2.2, the input vector x has 17 components, y represents the three time series for discharge, Calcium and Sodium, and z represents the corresponding field data. The function $f(\cdot)$ relating y to x develops as follows:

There is a fast f and a slow s sub-compartment for each of the three soil-type compartments DP, AS and GW. Updating the effective water stored from hour t to $t+1$ for

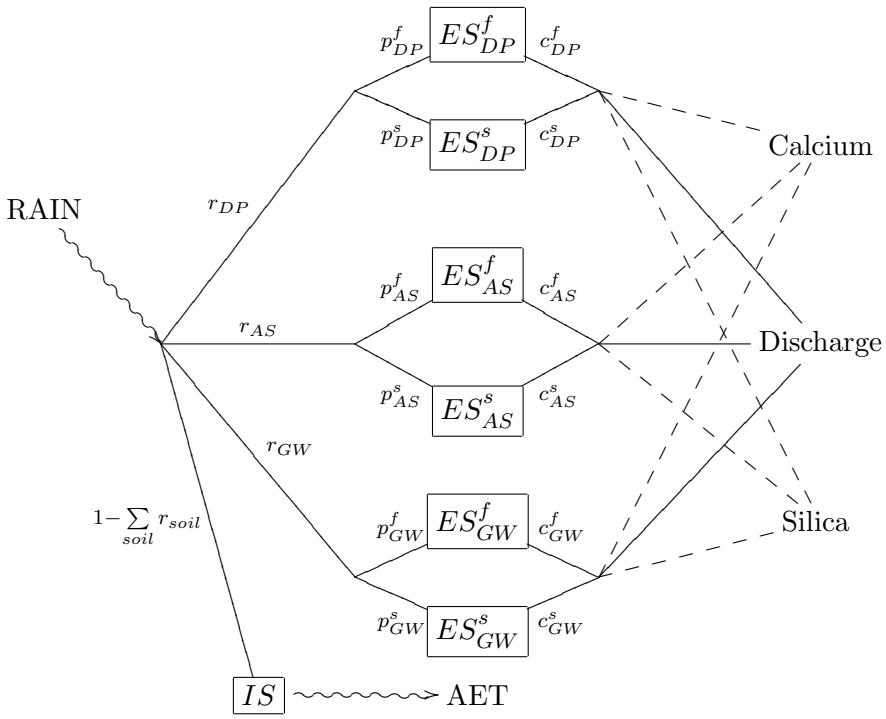


Figure 2: Three compartment rainfall runoff model.

each sub-compartment is governed by the equations

$$ES^f_{soil}(t+1) = ES^f_{soil}(t) + r_{soil}(t)p^f_{soil}\,\mathrm{RAIN}(t) - c^f_{soil}ES^f_{soil}(t) \tag{6}$$
$$ES^s_{soil}(t+1) = ES^s_{soil}(t) + r_{soil}(t)p^s_{soil}\,\mathrm{RAIN}(t) - c^s_{soil}ES^s_{soil}(t) \tag{7}$$

where *soil* is one of DP, AS and GW, $p^f_{soil} + p^s_{soil} = 1$

$$r_{soil}(t) = \frac{k_{soil}}{1 + \exp\left[a_{soil} - b_{soil}S(t)\right]} \tag{8}$$

with $k_{DP} + k_{AS} + k_{GW} = 1$ and $S(t)$, the total water stored in the system at time t is given by

$$S(t) = \sum_{soil}[ES^f_{soil}(t) + ES^s_{soil}(t)] + IS(t)$$

That is, the *total water storage* S in the system at any time is the sum of the *effective storages* for each soil type, both fast and slow, plus the overall residual *ineffective storage* IS. Physical interpretations of the six parameters for each compartment will emerge in the next subsection. Updating the total storage from t to $t+1$ is governed by the equation

$$S(t+1) = S(t) + \mathrm{RAIN}(t) - \mathrm{AET}(t) - \sum_{soil} F_{soil}(t)$$

where the $F_{soil}(t) = c^f_{soil}ES^f_{soil}(t) + c^s_{soil}ES^s_{soil}(t)$ are the *flows* out of each soil-type compartment. Similarly, updating the ineffective storage from t to $t+1$ is governed by the equation

$$IS(t+1) = IS(t) + \mathrm{RAIN}(t)[1 - \sum_{soil} r_{soil}(t)] - \mathrm{AET}(t) \tag{9}$$

Hourly model outputs, discharge $D(t)$, Calcium $Ca(t)$ and Silica $Si(t)$ are given by

$$D(t) = \sum_{soil} F_{soil}(t) \quad (10)$$

$$Ca(t) = \sum_{soil} T^{Ca}_{soil} F_{soil}(t)/D(t) \quad (11)$$

$$Si(t) = \sum_{soil} T^{Si}_{soil} F_{soil}(t)/D(t) \quad (12)$$

where the T^{tracer}_{soil} govern the tracer concentrations of Ca and Si emanating from each soil-type compartment.

Thus, to run the model $y = f(x)$ we need (i) a computer code implementation of $f(\cdot)$; (ii) valid values for the 17 components of x; (iii) the forcing functions RAIN and AET; (iv) the initial conditions ES^f_{soil}, ES^s_{soil} and IS at $t = 0$; and (v) the values of the six tracer concentrations T^{tracer}_{soil}.

4 Modular Dynamic Emulation

4.1 Dynamic Emulation

Recall that our rainfall runoff model outputs three time series over 839 time steps. However, we can regard the model as being generated by a single time-step simulator f that relates the state of the system at time $t-1$ to the state at time t. The single time-step simulator f is applied iteratively 839 times to produce the time series outputs of water discharge, Calcium and Silica. Emulating the single time-step simulator f is simpler than emulating the full 839-step simulator. Moreover, it will allow us to track various uncertainties through time, such as uncertainty in the rainfall forcing function and uncertain model discrepancy.

The simulator f has 26 inputs: the 17 usual inputs x to the model, the forcing functions $RAIN(t)$ and $A(t) \equiv AET(t)$ at time t and the 7 state variables $ES^{f/s}_i(t-1)$ and $S(t-1)$ at time $t-1$. The 7 State Variables $ES^{f/s}_i(t)$ and $S(t)$ are outputs of f at time t. Dynamic emulation of the single time-step simulator $f(x, A, S)$, where A and S represent forcing functions and state variables, respectively, is often not straightforward. Moreover, an accurate emulator for $f(x, A, S)$ may still perform poorly over the 839 time step iterations, due to the accumulation of emulator error. The 7 state variable inputs S have such a dominant effect on the outputs of $f(x, A, S)$, it can be hard to pick up subtle impact of the other inputs. For example, emulation techniques such as active variables can be misleading. Accounting for other sources of uncertainty, as required in reification, can still be difficult at this level, although the process is much easier than direct reification, see House (2009), as the modeller can consider improvements at the single time-step level.

4.2 Modular Emulation

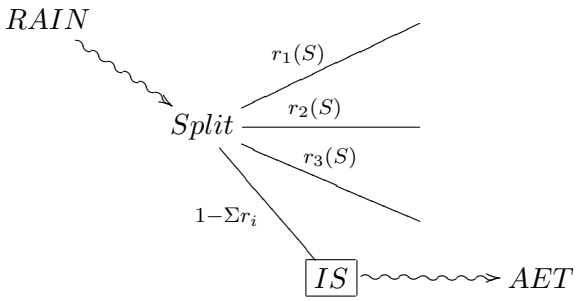
Dynamic emulation requires identification and access to the state variables of the model. If, in addition, we have access to some or all of the internal variables of the model, we can explore its internal processes and attempt "modular emulation". An advantage of modular emulation is that we may be able to emulate the more subtle parts of the model that may be masked by dynamic emulation. A further important advantage of modular emulation is that it's relatively simple to incorporate other sources of uncertainty when linking the model to reality. That is, it is easier for the expert to consider improvements to an individual process or module rather than to the whole model, thus simplifying

reification. A downside is the difficulty of combining the modular emulators together, as is the case with general dynamic emulation. To incorporate other sources of uncertainty in order to reify the current model, we require satisfactory modular emulators. It should be noted that one run of the rainfall runoff simulator gives 839 runs of the single time-step simulator f and each of the three modular simulators for the same cost.

We identified active variables for each of the three modules described below and constructed fast emulators composed of linear models with second-order polynomials in the active variables. The modular emulators were combined using the model structure and, where necessary, linearised approximations were used to calculate propagation of uncertainties. We intend to improve the emulators in a future account, for example, by using gaussian process error terms instead of assuming uncorrelated errors, as we have done here at this stage of development of modular emulation.

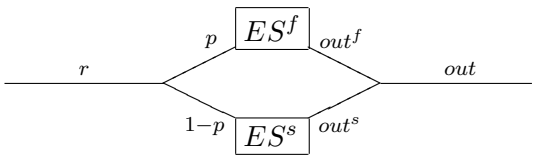
We now depict the three modules and indicate the emulation process we adopted for each of them.

4.3 Module 1



We found that the r_i depend on only k_i, a_i, b_i, S and RAIN and we were able to emulate the split function outputs r_i to a reasonable high degree of accuracy. We incorporated uncertainty into the forcing function RAIN, and will report on this elsewhere. In order to explore possible improvements to this module (see equations (2) and (3)), we have included correlated noise on the 4 outputs of the split function. This correlation results from the requirement that the four noise components sum to zero, which is equivalent to the conservation of water entering the system. We discuss the effects of this model improvement in Section 5.

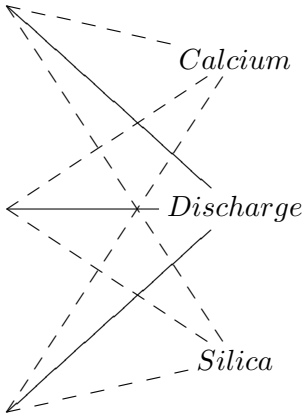
4.4 Module 2



Similarly, we can emulate and reify any one of the three 'compartments', as depicted above. We can choose how far 'into the box' we want to go; for example, we find that out^f at time t only depends on the two inputs $ES^f(t-1)$ and c^f, and is trivial to emulate.

Again, we can reify this compartment process, or parts of it, although we do not do so here.

4.5 Module 3



The final simulator outputs, Discharge, Calcium and Silica, depicted in the module above, are again straightforward to emulate. The runoff model assumes that the tracer concentrations of Calcium and Silica (which govern their concentrations in the runoff) are treated as fixed with known values. Modular emulation straightforwardly allows incorporation of tracer uncertainty, which is a first step in a reification of the runoff model. We analysed the effects of such uncertainty and discuss the results in Section 5.

To carry out any one of the above single time-step emulation processes (whether it be direct dynamic emulation or modular emulation) and track any uncertainty it was necessary to approximate various means, variances and covariances of certain functions of random quantities using first-order Taylor series approximations. These approximations are detailed in Appendix A.

4.6 Emulator Performance

Fig. 3 shows the emulator prediction for the 7 state variables $ES^f_{soil}(t)$, $ES^s_{soil}(t)$ and $S(t)$ over the complete time series of 839 hours, for a single run. The emulator mean is given by the blue line with a credible two-sigma interval given by the red lines. The actual run output is the black line. Note that the modular dynamic emulator behaves well for the first six state variables, but deviates from the true run output of the 7th state variable $S(t)$ at late times, after many steps. The performance is sufficient for our purposes of investigating internal model discrepancies. Similarly Fig. 4 shows the emulator prediction for the 7 state variables $ES^f_{soil}(t)$, $ES^s_{soil}(t)$ and $S(t)$ from 550 to 650 hours, for a single run. This is a zoomed in version of Fig. 3. Note that the adjusted R^2 for each of the state variables is in excess of 0.93, suggesting relatively accurate emulation.

5 Towards Reification: Examples

We explore two types of internal discrepancy. The first involves uncertainty regarding the propagation of the state vector, resulting from the addition of noise to the split functions

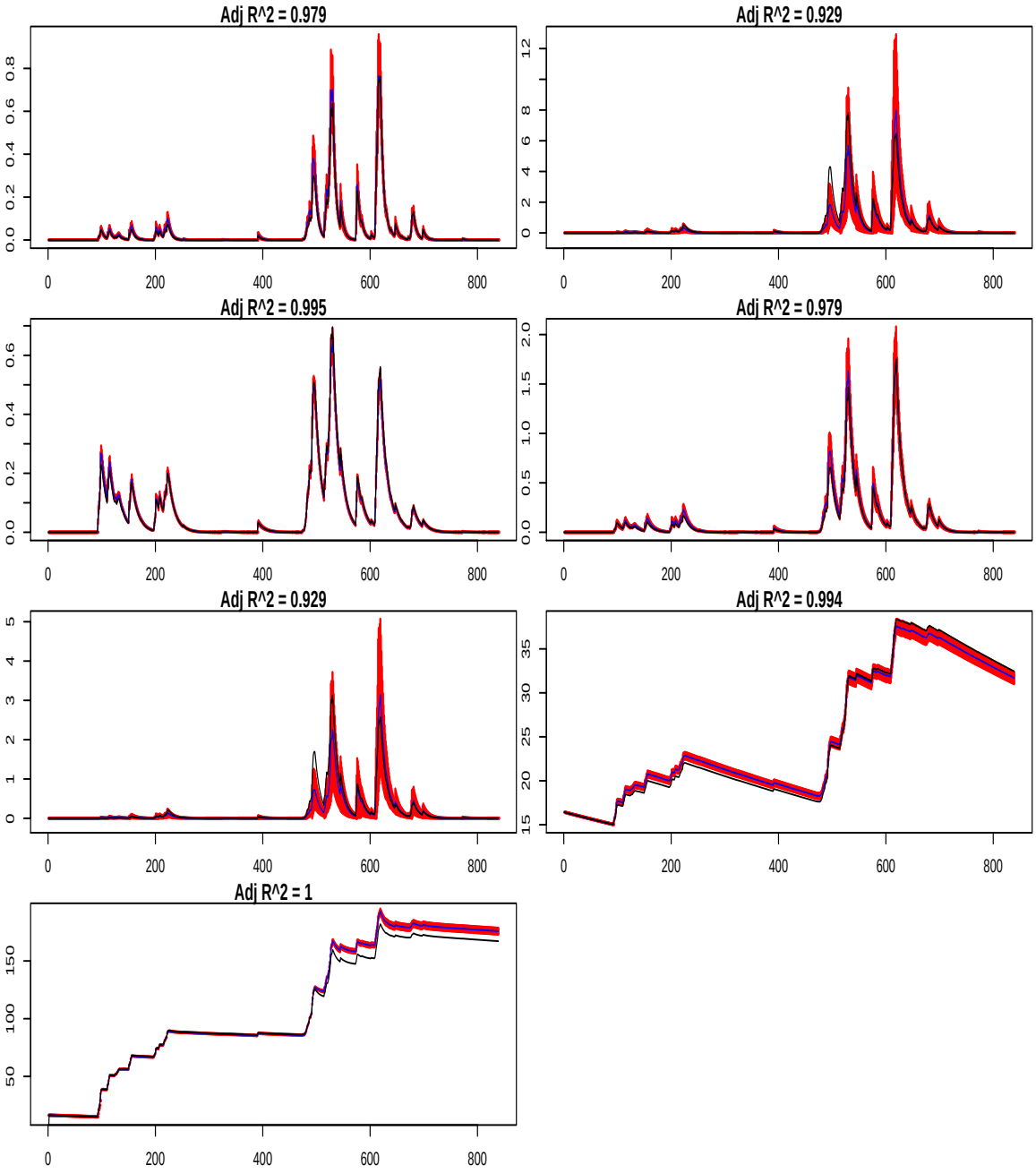


Figure 3: Emulator prediction for the 7 state variables $ES^f_{soil}(t)$, $ES^s_{soil}(t)$ and $S(t)$ over the complete time series of 839 hours, for a single run. The emulator mean is given by the blue line with a credible two-sigma interval given by the red lines. The actual run output is the black line. Note that the modular dynamic emulator behaves well for the first six state variables, but deviates from the true run output of the 7th state variable $S(t)$ at late times, after many steps. The performance is sufficient for our purposes of investigating internal model discrepancies.

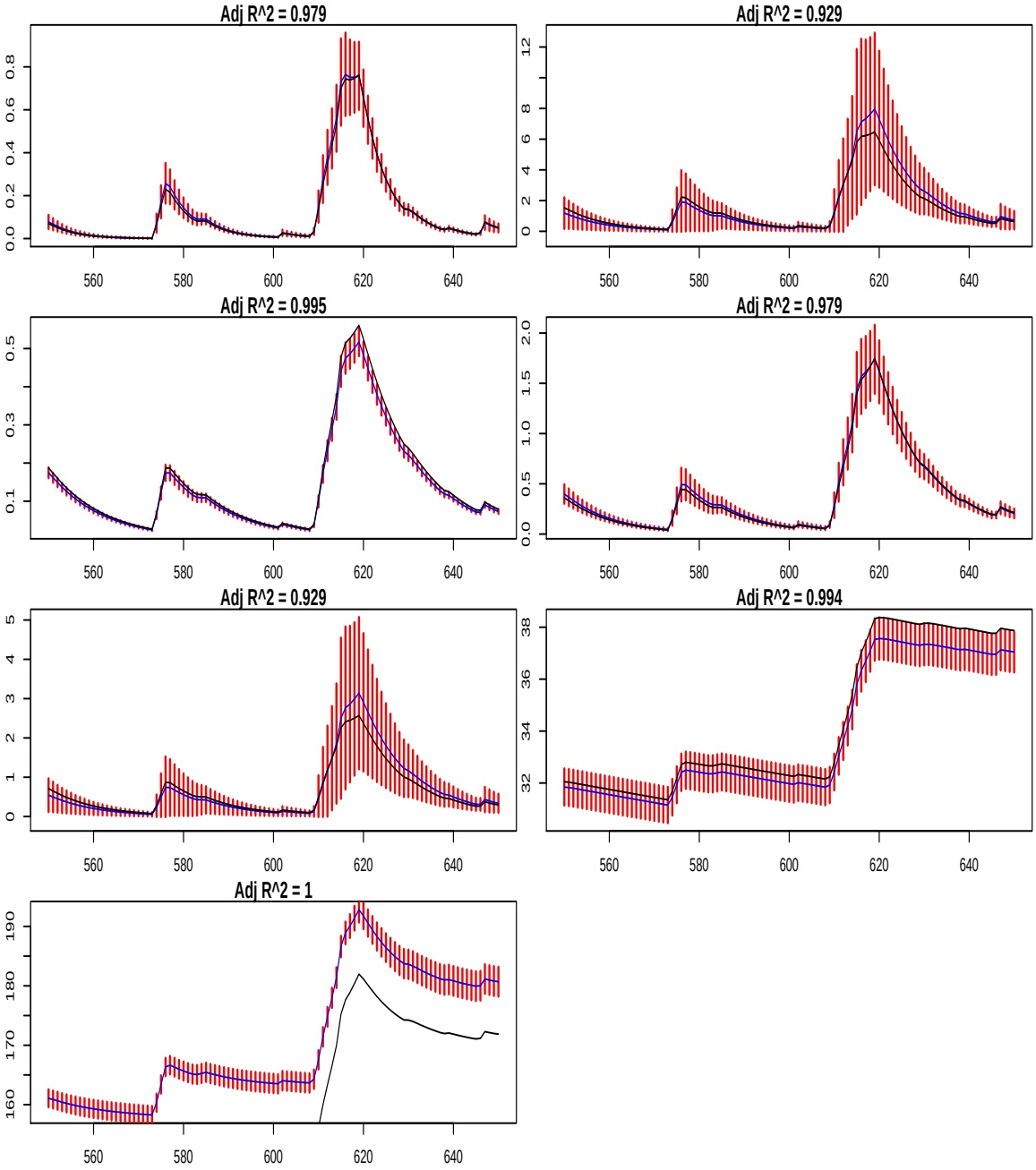


Figure 4: Emulator prediction for the 7 state variables $ES^f_{soil}(t)$, $ES^s_{soil}(t)$ and $S(t)$ from 550 to 650 hours, for a single run. The emulator mean is given by the blue line with a credible two-sigma interval given by the red lines. The actual run output is the black line.

discussed in Section 4.3. The second seeks to incorporate uncertainty on the tracer concentrations, previously assumed constant. Each of these would form part of a structural reification process discussed in Section 2.1, which we postpone to a future account.

5.1 Effects on State Variables

The effect of introducing uncertainty on the propagation of the state vector can be seen in Fig. 5 and Fig. 6.

Fig. 5 shows emulator prediction for the 7 state variables $ES^f_{soil}(t)$, $ES^s_{soil}(t)$ and $S(t)$ over the complete time series of 839 hours, for a single run, including the state propagation model discrepancy. The emulator mean is given by the blue line with a credible two-sigma interval given by the red lines. The actual run output is the black line. This state propagation uncertainty is addressed by adding noise to each output of the split function of module 1 depicted in Section 4.3. This is done by inflating each of the σ_i in equation (14) of Appendix A by 30 percent, which effectively increases the uncertainty of the split function's six outputs which multiplies the incoming rainfall: see the second term on the right hand side of equations (6) and (7). This represents the fact that module 1 is the most uncertain part of the model, and the process we use to capture this uncertainty can be viewed as a generalised reification step. These results should be compared to those shown Fig. 3.

Fig. 6 shows a zoomed in version of Fig. 5, covering hours 550 to 650 and should be compared with Fig. 4.

5.2 Effects on Calcium Output

We now consider the effect of introducing uncertainty in the propagation of the state vector on the calcium output, shown in panels 1 and 3 of Fig. 7, and of introducing uncertainty in both the propagation of the state vector and on the tracer concentrations for calcium output, shown in panels 2 and 4 of Fig. 7.

Fig. 7 shows the emulator prediction for Calcium $Ca(t)$ (see equation (11)) for a single run including state propagation discrepancy over the full time series, with and without tracer uncertainty (panels 1 and 2), and over a reduced part of the time series (panels 3 and 4). The green lines show the proportion of the $Ca(t)$ output uncertainty due to the added 10 percent uncertainty on each of the tracer concentrations T^{Ca}_{soil} (see equation (11)), compared to the state propagation uncertainty and emulator uncertainty in red. In dry periods the Calcium tracer uncertainty is far more important, while in wet periods the state propagation uncertainty dominates.

6 Conclusions

We have developed fast modular dynamic emulators of sufficient accuracy for exploration of internal model discrepancies, which may form part of a reification analysis. We have incorporated uncertainty due to the tracer concentrations in the runoff model and due to uncertain state propagation. There remains much to investigate, such as full structural reification on the dynamic or modular level; for example, including an additional compartment, output functions, split function. In a future account we will consider: checking assumptions via simulation; comparison of direct emulation with dynamic and modular emulation; assessment of the relative impact of different uncertainties, upgrading constant inputs to time dependent processes and external reification.

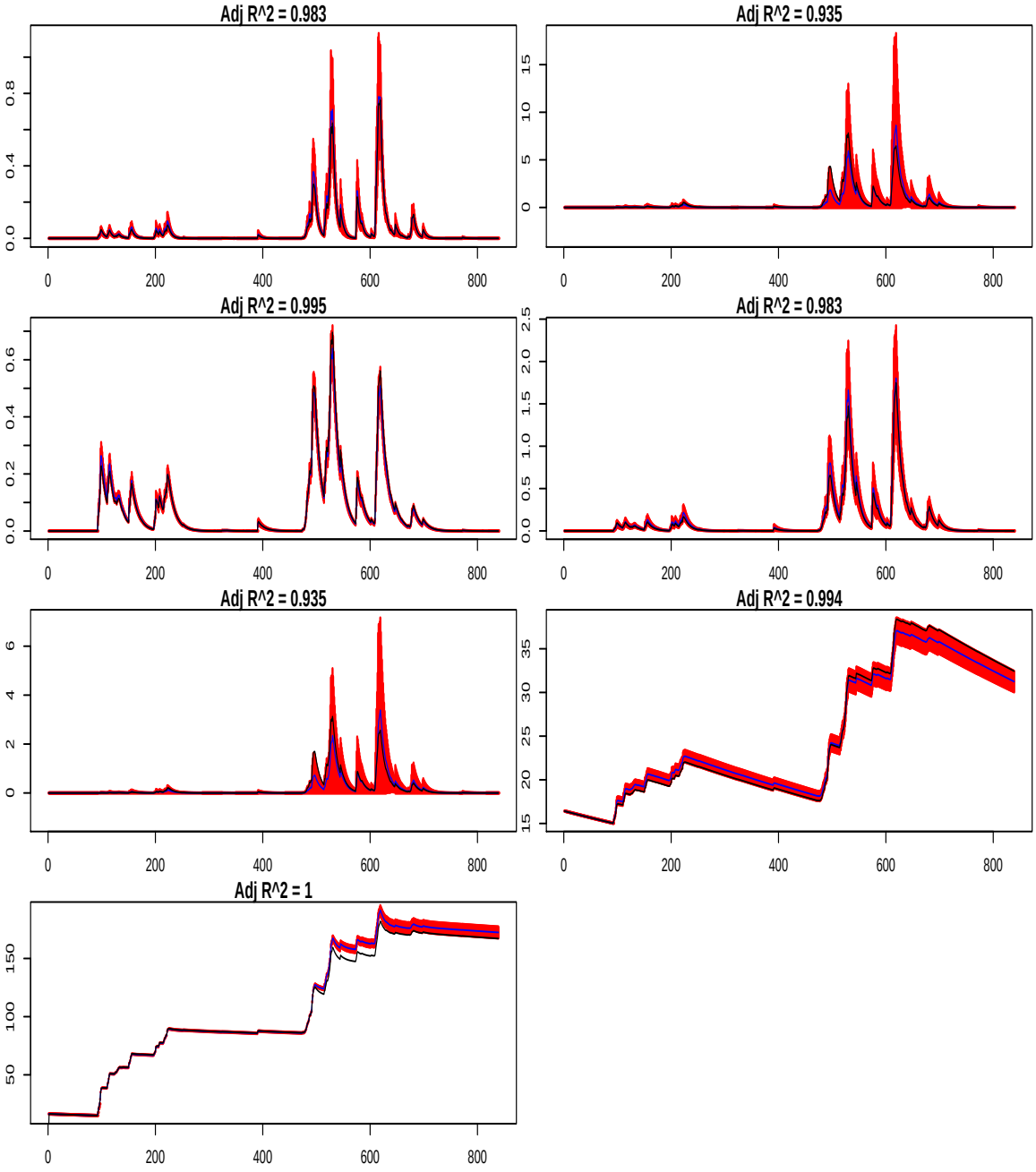


Figure 5: Emulator prediction for the 7 state variables $ES^f_{soil}(t)$, $ES^s_{soil}(t)$ and $S(t)$ over the complete time series of 839 hours, for a single run, including the state propagation model discrepancy. The emulator mean is given by the blue line with a credible two-sigma interval given by the red lines. The actual run output is the black line. This state propagation uncertainty is addressed by adding noise to each output of the split function of module 1 depicted in Section 4.3. This is done by inflating each of the σ_i in equation (14) of Section A by 30 percent, which effectively increases the uncertainty of the split function's six outputs which multiplies the incoming rainfall: see the second term on the right hand side of equations (6) and (7). This represents the fact that module 1 is the most uncertain part of the model, and the process we use to capture this uncertainty can be viewed as a generalised reification step. These results should be compared to those shown Fig. 3.

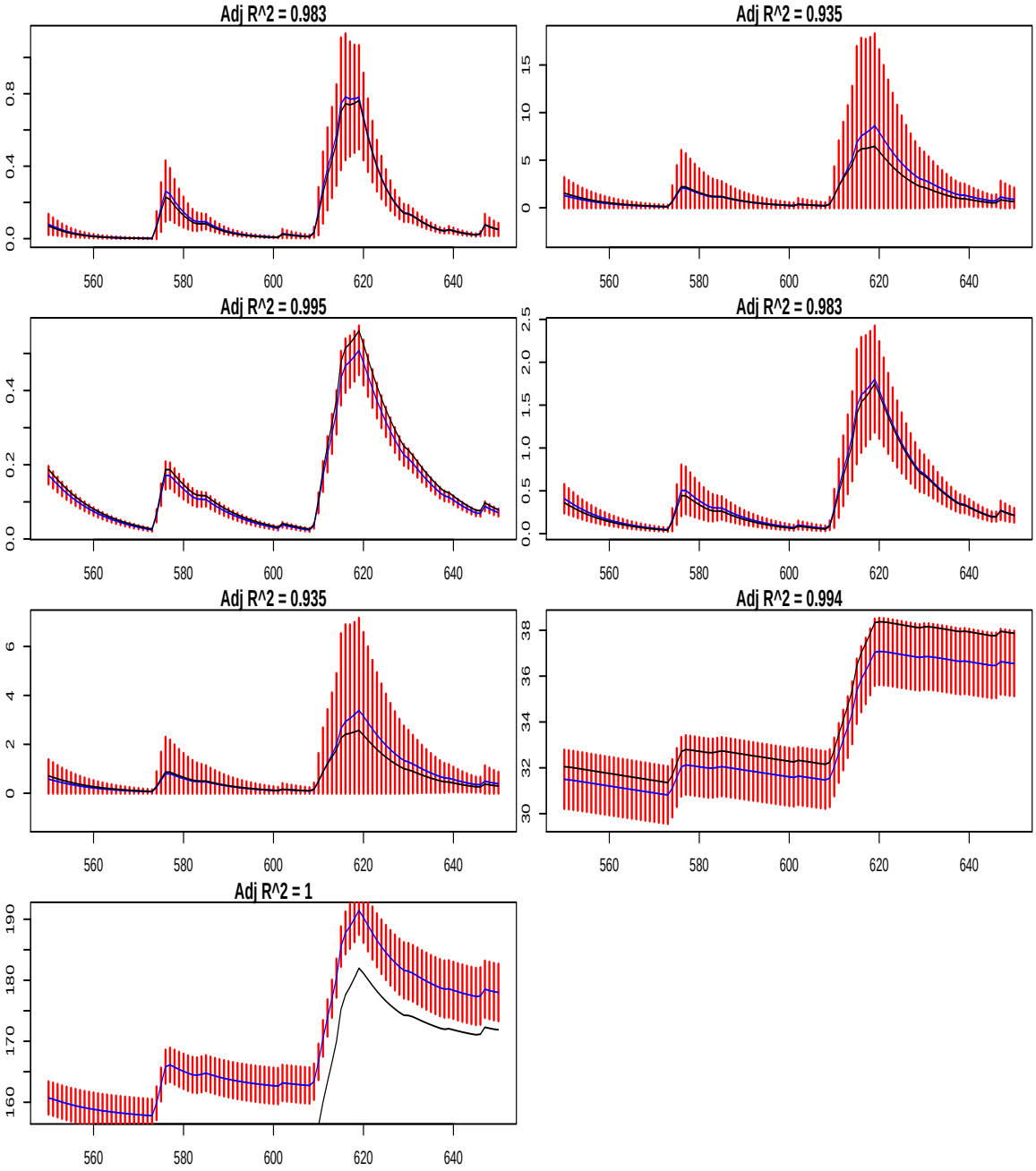


Figure 6: Emulator prediction for the 7 state variables $ES^f_{soil}(t)$, $ES^s_{soil}(t)$ and $S(t)$ from 550 to 650 hours, for a single run, including the state propagation model discrepancy. The emulator mean is given by the blue line with a credible two-sigma interval given by the red lines. The actual run output is the black line. This state propagation uncertainty is addressed by adding correlated noise to the output of the split function of module 1 depicted in Section 4.3. These results should be compared to those shown in Fig. 4.

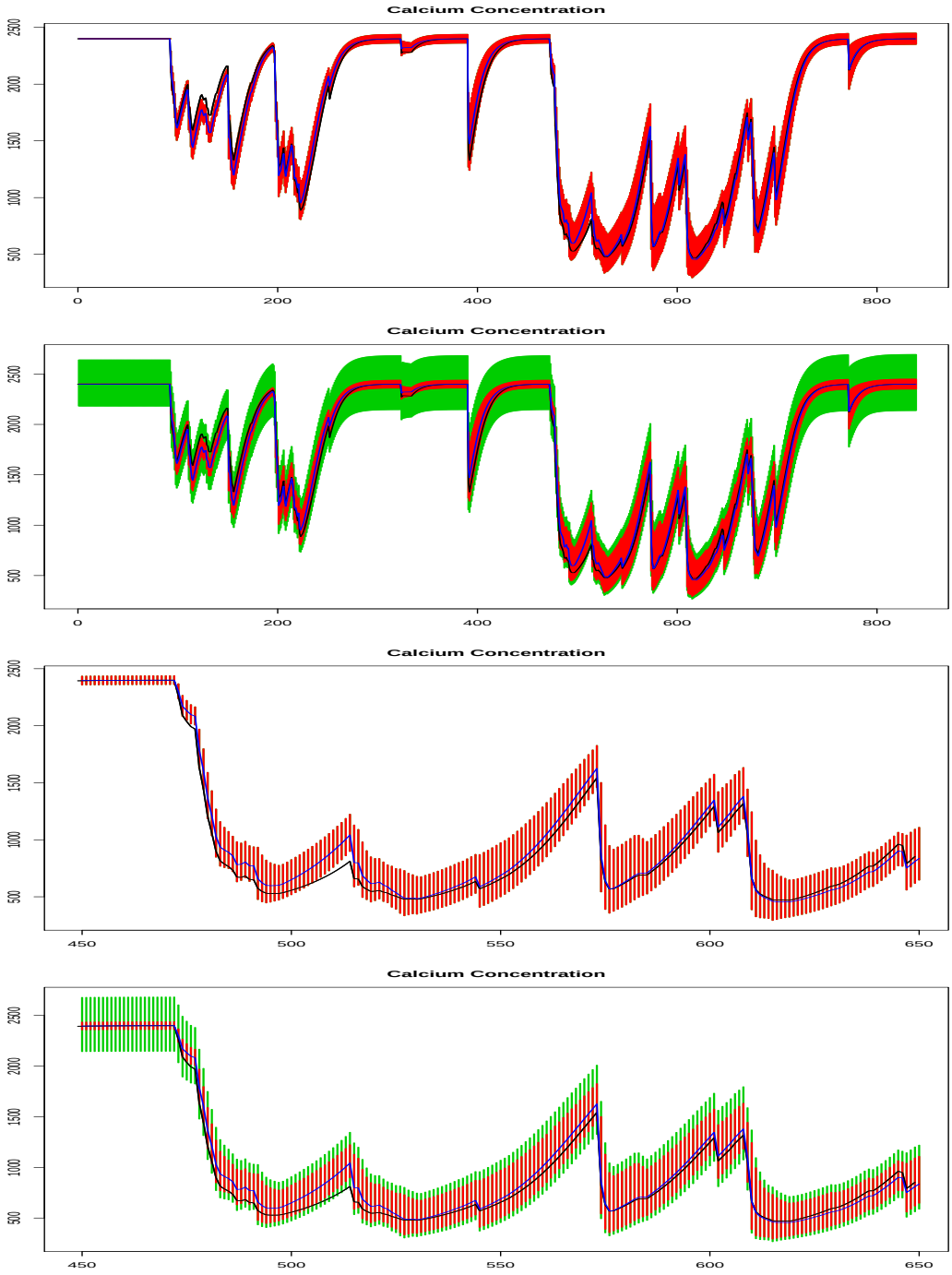


Figure 7: Emulator prediction for Calcium $Ca(t)$ (see equation (11)) for a single run including state propagation discrepancy over the full time series, with and without tracer uncertainty (panels 1 and 2), and over a reduced part of the time series (panels 3 and 4). The green lines show the proportion of the $Ca(t)$ output uncertainty due to the added 10 percent uncertainty on each of the tracer concentrations T_{soil}^{Ca} (see equation (11)), compared to the state propagation uncertainty and emulator uncertainty in red. In dry periods the Calcium tracer uncertainty is far more important, while in wet periods the state propagation uncertainty dominates.

Acknowledgements

We are extremely grateful to Keith Beven for many helpful discussions on the runoff model and its application to the Haute Mentue catchment and to Leanna House for explaining the model, providing her `R` code and for valuable discussions on dynamic emulation.

This internal report was produced with the support of the Basic Technology Initiative as part of the "Managing Uncertainty for Complex Models" project and an EPSRC Mobility Fellowship for Ian Vernon.

References

Craig, P. S., Goldstein, M., Rougier, J. C., and Seheult, A. H. (2001), "Bayesian forecasting for complex systems using computer simulators," *Journal of the American Statistical Association*, 96, 717–729.

Craig, P. S., Goldstein, M., Seheult, A. H., and Smith, J. A. (1997), "Pressure matching for hydrocarbon reservoirs: a case study in the use of Bayes linear strategies for large computer experiments," in *Case Studies in Bayesian Statistics*, eds. Gatsonis, C., Hodges, J. S., Kass, R. E., McCulloch, R., Rossi, P., and Singpurwalla, N. D., New York: Springer-Verlag, vol. 3, pp. 36–93.

Goldstein, M. and Rougier, J. C. (2009), "Reified Bayesian modelling and inference for physical systems (with Discussion)," *Journal of Statistical Planning and Inference*, 139, 1221–1239.

House, L. (2009), "Application of Reification to a Rainfall-Runoff Computer Model," *MUCM Internal Report 2.1.4.*

Iorgulescu, I., Beven, K. J., and Musy, A. (2005), "Data-based modelling of runoff and chemical tracer concentrations in the Haute-Mentue research catchment (Switzerland)," *Hydrological Processes*, 19, 2557–2573.

Kennedy, M. C. and O'Hagan, A. (2001), "Bayesian calibration of computer models," *Journal of the Royal Statistical Society, Series B*, 63, 425–464.

MUCM (2009), *MUCM Toolkit Release 3*, Aston, England.

O'Hagan, A. (2006), "Bayesian analysis of computer code outputs: A tutorial," *Reliability Engineering and System Safety*, 91, 1290–1300.

Rougier, J. (2009), "Formal Bayes methods for model calibration with uncertainty," in *Applied Uncertainty Analysis for Flood Risk Management*, eds. Beven, K. and Hall, J., Imperial College Press / World Scientific.

A *Approximate moments of a single time-step runoff model emulator*

The "split" functions for the runoff model are of the form

$$W_i = k_i(S)h(\epsilon_i) \tag{13}$$

where

$$k_i(S) = \exp\left[a_i + b_i g(S) + c_i g^2(S)\right] \qquad h(\epsilon_i) = \exp\left[\epsilon_i\right] \tag{14}$$

S and the ϵ_i are uncorrelated, with $\mathrm{E}[S] = \mu$, $\mathrm{Var}[S] = \sigma^2$, $\mathrm{E}[\epsilon_i] = 0$ and $\mathrm{Var}[\epsilon_i] = \sigma_i^2$; and current choices for $g(S)$ are S and $\log S$.

We use the first-order approximation $f(X) = f(\mu)+(X-\mu)f^{'}(\mu)$ to derive the following approximate moments of $f(X)$ given $\mathrm{E}[X] = \mu$ and $\mathrm{Var}[X] = \sigma^2$

$$\mathrm{E}[f(X)] = f(\mu) \qquad \mathrm{Cov}[f_i(X), f_j(X)] = \sigma^2 f_i^{'}(\mu) f_j^{'}(\mu) \tag{15}$$

and hence, $\mathrm{Var}[f(X)] = \sigma^2 f^{'}(\mu)^2$.

Straightforward calculations show that approximately $\mathrm{E}[W_i] = k_i(\mu)$ and

$$\mathrm{Cov}[W_i, W_j] = k_i^{'}(\mu)k_j^{'}(\mu)\sigma^2 + \delta_{ij}[k_i(\mu)k_j(\mu) + k_i^{'}(\mu)k_j^{'}(\mu)\sigma^2]\sigma_i\sigma_j \tag{16}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. In particular,

$$\mathrm{Var}[W_i] = k_i^{'}(\mu)^2\sigma^2 + [k_i(\mu)^2 + k_i^{'}(\mu)^2\sigma^2]\sigma_i^2 \tag{17}$$

This variance is based on the following general result for two uncorrelated random quantities X and Y

$$\mathrm{Var}[XY] = \mathrm{Var}[X][\mathrm{E}[Y]]^2 + \mathrm{E}[X^2]\mathrm{Var}[Y] \tag{18}$$

We can express these expectation, variance and covariance formulae for the W_i in the vector form

$$\mathrm{E}[W] = k(\mu) \qquad \mathrm{Var}[W] = \sigma^2 k^{'}(\mu)k^{'}(\mu)^T + D \tag{19}$$

where D is a diagonal matrix with entries

$$D_{ii} = [k_i(\mu)^2 + k_i^{'}(\mu)^2\sigma^2]\sigma_i^2 \tag{20}$$

Note that

$$k_i^{'}(\mu) = [b_i + 2c_i g(\mu)]g^{'}(\mu)k_i(\mu) \tag{21}$$

Including a random forcing function

When we include a forcing function F, such as *rainfall* or AET in the regression which is measured with error, (14) above is modified to be

$$k_i(S) = \exp\left[a_i + b_i g(S) + c_i g^2(S) + d_i F\right] \quad h(\epsilon_i + d_i\varepsilon) = \exp\left[\epsilon_i + d_i\varepsilon\right] \tag{22}$$

where ϵ_i is as before, d_i is the regression coefficient for the forcing function F and ε is the error in measuring F.

Putting $\delta_i = \epsilon_i + d_i\varepsilon$, we see that $\mathrm{E}[\delta_i] = 0$, $\mathrm{Var}[\delta_i] = \sigma_i^2 + d_i^2\sigma_\varepsilon^2$ and $\mathrm{Cov}[\delta_i,\, \delta_j] = d_i d_j \sigma_\varepsilon^2$.

It then follows, using similar approximations to those above, that

$$\begin{aligned}\mathrm{Cov}[W_i, W_j] = k_i^{'}(\mu)k_j^{'}(\mu)\sigma^2[1 + d_i d_j \sigma_\varepsilon^2] + d_i d_j k_i(\mu)k_j(\mu)\sigma_\varepsilon^2 + \\ \delta_{ij}[k_i(\mu)k_j(\mu) + k_i^{'}(\mu)k_j^{'}(\mu)\sigma^2]\sigma_i\sigma_j\end{aligned} \tag{23}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. In particular,

$$\mathrm{Var}[W_i] = k_i^{'}(\mu)^2\sigma^2[1 + d_i^2\sigma_\varepsilon^2] + d_i^2 k_i(\mu)^2\sigma_\varepsilon^2 + [k_i(\mu)^2 + k_i^{'}(\mu)^2\sigma^2]\sigma_i^2 \tag{24}$$

Writing $k(\mu)$ as the vector with i-th element $k_i(\mu)$, $dk(\mu)$ as the vector with i-th element $d_i k_i(\mu)$, and $dk^{'}(\mu)$ as the vector with i-th element $d_i k_i^{'}(\mu)$, we can express these expectation, variance and covariance formulae for the W_i in the vector form $\mathrm{E}[W] = k(\mu)$ and

$$\mathrm{Var}[W] = \sigma^2 k^{'}(\mu)k^{'}(\mu)^T + [\sigma^2 dk^{'}(\mu)dk^{'}(\mu)^T + dk(\mu)dk(\mu)^T]\sigma_\varepsilon^2 + D \tag{25}$$

where D is a diagonal matrix with entries

$$D_{ii} = [k_i(\mu)^2 + k_i^{'}(\mu)^2\sigma^2]\sigma_i^2 \tag{26}$$

which is the same as (20).