FOUNDATIONS OF VERIFICATION AND VALIDATION
A LOGICAL DERIVATION FROM THE SCIENTIFIC THEORY OF TRUTH

J. M. Smith
www.compuscience.uk

Summary

Verification and validation in engineering simulation is an emerging subject within which the primary definitions and taxonomies remain topics of debate.

Authors have proposed various definitions, component taxonomies and frameworks for the simulation process and associated verification and validation tests, e.g. [1]. However, these schemes have invariably been derived from empirical or ad-hoc bases.

This paper presents a unique analytical approach for deducing the framework and taxonomy of simulation process components, error sources and associated verification and validation processes from a logical base in the scientific theory of truth.

This paper will show that casting a simulation model in syllogistic form renders it amenable to error testing by the well-established scientific truth theories of coherence and correspondence [2]. A logically structured simulation framework and taxonomy of components and error sources naturally emerges from this process.

This approach both clarifies the simulation verification and validation processes by revealing the logical structure of their underlying simulation components and error sources, and legitimises them as forms of coherence and correspondence truth tests.

Introduction

The process of deriving a framework and component taxonomy for the engineering simulation process begins by casting the simulation model in syllogistic form. The truth tests of coherence and correspondence are then applied to this syllogism. Internal relations within the model are tested by the coherence theory, and external relations (real-world) are tested by the correspondence theory.

Applying this approach, a logical framework of simulation components and error source categories naturally emerges, where the principal taxonomic components can be identified as the idealised model and the computational model.

Moreover, applying the coherence test to the computational model is recognised as the ‘verification’ process, and applying the correspondence test between the idealised model and a corroborative system (e.g. experimental measurements) is recognised as the ‘validation’ process.

The unique advantage of the approach taken in this paper is that it is analytical rather than empirical, and delivers a logical taxonomic structure of simulation components, error sources and associated verification and validation processes from a sound base in logic and the scientific theory of truth.

1. Scientific explanation and the syllogism

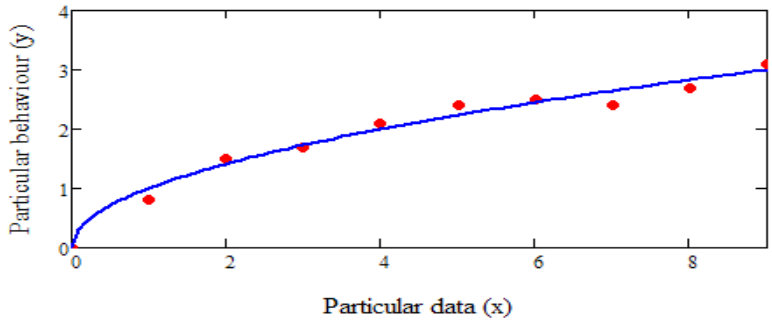
A syllogistic argument is a formal deductive argument consisting of a major premise, a minor premise and a conclusion, which states that: if p implies q, and q implies r, then p implies r:

$$\underbrace{[p \to q]}_{\substack{major\\premise}} \wedge \underbrace{[q \to r]}_{\substack{minor\\premise}} \underset{deduction}{\Rightarrow} \underbrace{[p \to r]}_{conclusion} \qquad (1)$$

The crucial feature of this syllogistic argument is that if the premises $p \to q$ and $q \to r$ are true and the logic relating them is correctly processed, then the conclusion $p \to r$ will necessarily be true. Conversely, if one of more of these premises is false, or the logic is incorrectly processed, then the conclusion will be false.

It is apparent that the syllogism possesses a predictive capability, however this is merely a tautological inevitability which follows from the premises and logic of $p \to q$ and $q \to r$, which already contains the conclusion $p \to r$ within its range.

The potential for applying a syllogistic argument to represent a scientific model is suggested by Hempel's model of scientific explanation [4], which states that scientific explanations can be cast in the form of a logical argument consisting of a general laws and particular facts.



experimental measurements	:	**general behaviour**	$\underset{induction}{\Leftarrow}$	**particular data**	$\wedge$	**particular behaviour**	
$\Downarrow$ *idealisation*		$\Downarrow$ *idealisation*		$\Downarrow$ *idealisation*		$\Uparrow$ *predicion*	
idealised model	:	$\underbrace{[y = \sqrt{x}]}_{\substack{mathematical\\model}}$	$\wedge$	$\underbrace{[\sqrt{x} = \sqrt{2}]}_{\substack{parametric\\data}}$	$\underset{deduction}{\Rightarrow}$	$\underbrace{[y = \sqrt{2}]}_{solution}$	(2)

Figure 1 Experimental measurements idealised as a predictive syllogistic model

Clearly, the real-world is not naturally organised into general laws (e.g. behaviours) and particular facts (e.g. data), but experimental measurements of physical behaviour in response to particular loading data for example, enables us to inductively infer empirical laws which describe the general behaviour of the system.

For example, based on some experimental measurements of a real-world system, it is inferred that the general behaviour of the system (the dots on the graph in Fig.1) can be idealised by the mathematical function $f(x) = y - \sqrt{x}$ (the line on the graph and upper row in Fig.1).

Applying Hempel's model, this general behaviour together with particular data (e.g. $x = 2$), enables us to predict the particular behaviour of the system at $x = 2$ by constructing a syllogistic argument with $\left[y = \sqrt{x}\right]$ as the major premise and $\left[\sqrt{x} = \sqrt{2}\right]$ as the minor premise. Processing this syllogism yields the logical conclusion that $\left[y = \sqrt{2}\right]$, as shown in Eq. (2) in Fig. 1.

It is noted that in practice we typically use numerical methods to compute our solutions by approximating the model using n discrete terms. This numerical approximation can be expressed within the syllogism as:

$$\underbrace{\left[y = \sqrt{x}\right]}_{\substack{numerical \\ model}} \wedge \underbrace{\left[\sqrt{x} = 1.41 \ldots n\right]}_{\substack{discrete \\ parametric\ data}} \underset{computation}{\Rightarrow} \underbrace{\left[y \underset{n\to\infty}{\to} \sqrt{2}\right]}_{\substack{discrete \\ solution}} \qquad (3)$$

The discrete nomenclature in (3) indicates that the deductive process in (2) has been transformed into a computational process whereby a diminution of computational error is now required to restore its deductive characteristic in the limit. In this sense, the numerical and mathematical models also converge in the limit, i.e. the approximate discrete solution of the numerical model in (3) converges towards the analytical solution of the mathematical model in (2) as the number of terms n in the numerical approximation is increased.

2. Scientific theories of truth

"A judgment is said to be true when it conforms to the external reality", Thomas Aquinas, c.1250.

This syllogistic form of the idealised model renders it amenable to testing by the well-established scientific truth theories of coherence and correspondence [2], where coherence theory tests the consistency of a statement's logic, and correspondence theory tests the conformity of a statement's premises and conclusions with real-world facts. The two terms are commonly defined as:

coherence*: logical consistency;*
correspondence*: conformance to fact.*

When we use idealised models to predict the behaviour of a real-world system, we make idealisation assumptions that introduce idealisation error. In addition, when we compute the

solution using numerical methods, we introduce computation error. These intrinsic sources of error are addressed by the scientific truth tests of correspondence and coherence respectively, as explained in the following sections.

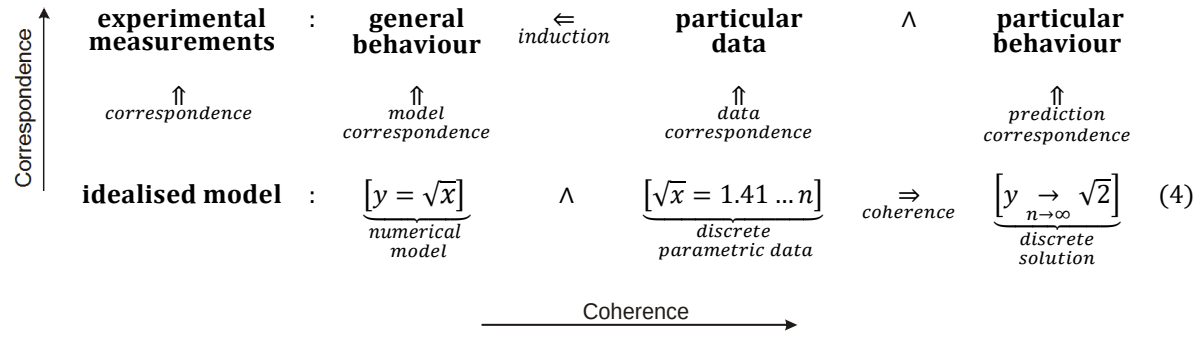
2.1 Coherence test for computation error

Coherence theory tests the consistency of a statement's logic. Because the logic of the syllogistic argument is sound, the computational model is in principle coherent, but numerical statements such as (3) require a diminution of computation error to realise that coherence in the limit (see Fig. 2). Hence, the coherence testing process addresses computation error, which can arise both in the numerical approximation and the calculation.

Once a coherently converged computational model has been achieved through diminution of computational error, the solution should then correspond with increasing accuracy to the experimental measurements as any remaining idealisation errors are eliminated (discussed in Section 2.2 below). Coherence testing is a prerequisite to correspondence testing.

2.2 Correspondence test for idealisation error

Correspondence theory tests the conformity of a statement's premises and conclusions with real-world facts. The idealised model, comprising a coherent mathematical model, data and solution, should correspond to the general behaviour, particular data and particular behaviour of the associated experimental measurements with increasing accuracy as the idealisation errors are identified and eliminated. Hence, the correspondence testing process addresses the idealisation error arising in the major premises (model error), minor premises (data error) and the ensuing conclusions (prediction error), as shown in Fig. 2.



(4)

Figure 2 The conclusion of an idealised model should correspond with increasing accuracy to the experimental measurements as the idealisation and computation error is reduced. In this figure, coherence is characterised by convergence within rows, and correspondence by convergence within columns.

3. Sources of error

The following example shows how the sources of error in the syllogistic modelling process can be identified by applying the two scientific tests for truth, where:

a) correspondence test – identifies idealisation error, consisting of model error and data error (see Section 2.2);
b) coherence test – identifies computation error, consisting of approximation error and calculation error (see Section 2.1).

3.1 Idealisation error – identified by the correspondence test

Fig. 3 shows a situation where reality behaves according to $[y = \sqrt{x}]$, and where the particular point of interest is $x = 2$. Writing this model in syllogistic form gives:

$$\underbrace{[y = \sqrt{x}]}_{\substack{mathematical \\ model}} \wedge \underbrace{[\sqrt{x} = \sqrt{2}]}_{\substack{parametric \\ data}} \underset{deduction}{\Rightarrow} \underbrace{[y = \sqrt{2}]}_{solution} \tag{5}$$

which, in words, means that if $[y = \sqrt{x}]$ and $[\sqrt{x} = \sqrt{2}]$ then $[y = \sqrt{2}]$.

However, let us assume that the actual solution function $[y = \sqrt{x}]$ is unknown, and that the analyst only observes the system behaviour in the interval $[0.5 < x < 1.25]$. In this case, the analyst believes that an alternative function, $[y = log(x) + 1]$, describes the unknown solution function within this interval, as shown in Fig. 3.

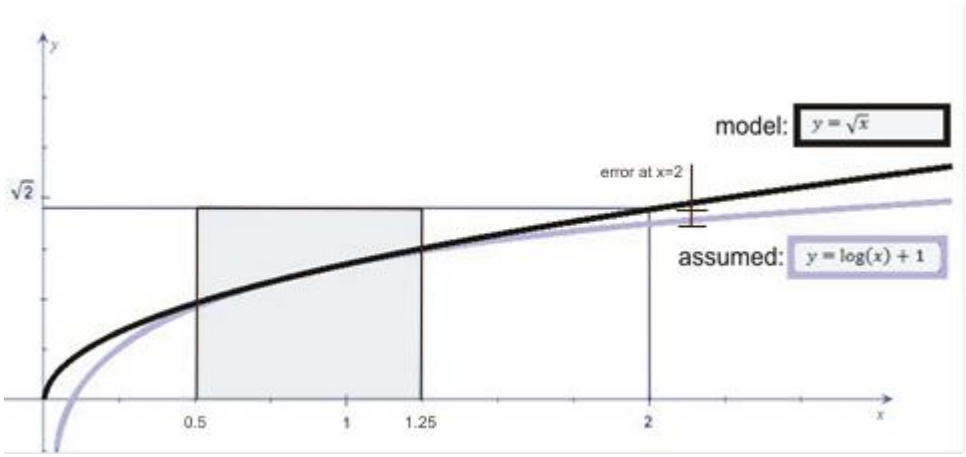


Figure 3 An alternative model $[y = log(x) + 1]$ *is sufficiently accurate within the interval* $[0.5 < x < 1.25]$

3.1.1 Model error

The first premise of the syllogistic argument (5) represents the mathematical model. It can be seen from Fig. 3 that while the model $[y = log(x) + 1]$ may be considered accurate within the interval $[0.5 < x < 1.25]$, the solution at the point $x = 2$ may fall outside the accepted range of accuracy.

$$\underbrace{[y = log(x) + 1]}_{\substack{mathematical \\ model}} \wedge \underbrace{[log(x) + 1 = log(2) + 1]}_{\substack{parametric \\ data}} \underset{deduction}{\Rightarrow} \underbrace{[y = log(2) + 1]}_{solution} \quad (6)$$

This type of idealisation error is caused by the use of an invalid mathematical model $[y = log(x) + 1]$ and is illustrated in the function range plot shown in Fig. 3.1a.

3.1.2 Data error

The second premise in (5) represents the model's data, and if that is false, say $x = 9$ (rather than the correct value, $x = 2$), then the conclusion will again be false as shown in (7).

$$\underbrace{[y = \sqrt{x}]}_{\substack{mathematical \\ model}} \wedge \underbrace{[\sqrt{x} = \sqrt{9}]}_{\substack{parametric \\ data}} \underset{deduction}{\Rightarrow} \underbrace{[y = 3]}_{solution} \quad (7)$$

This type of idealisation error is caused by invalid data and is illustrated in the function range plot shown in Fig. 3.1b.

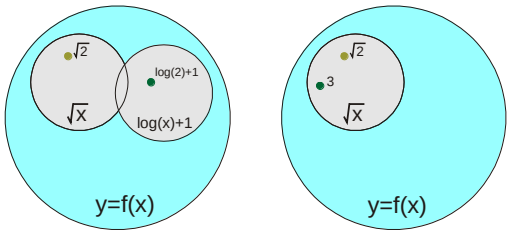


Figure 3.1 Idealisation error: a) model error; b) data error

In summary, the correspondence test addresses idealisation error which can arise in either of the two premises – error arising in the major premise is interpreted as model error, and error in the minor premise is interpreted as parametric data error.

3.2 Computation error – identified by the coherence test

Mathematical models cannot typically be solved analytically due to their complexity, therefore an approximate numerical model of the mathematical model is constructed which can be solved computationally. This process however, introduces both numerical approximation error and calculation error.

3.2.1 Approximation error

To illustrate approximation error we use a discrete data laden model. In this case, assuming that the logic of the argument will be correctly processed, the conclusion in (8) will still only be true within accepted bounds of numerical accuracy.

$$\underbrace{[y=\sqrt{x}]}_{\substack{numerical\\model}} \wedge \underbrace{[\sqrt{x}=1.41\ldots n]}_{\substack{discrete\\parametric\ data}} \underset{computation}{\Rightarrow} \underbrace{[y \underset{n\to\infty}{\to} \sqrt{2}]}_{\substack{discrete\\solution}} \tag{8}$$

This type of computation error is caused by approximating the mathematical model as a discrete, data laden numerical model, and is illustrated in the function range plot shown in Fig. 3.2a. In this case the discrete solution approaches the analytical solution as the number of decimal digits n approaches infinity.

3.2.2 Calculation error

To illustrate calculation error, we assume that the numerical model with its discrete data represent the mathematical model and its data with sufficient accuracy. However, in this example the conclusion in (9) is false because the arithmetical logic of the argument has been violated.

$$\underbrace{[y=\sqrt{x}]}_{\substack{mathematical\\model}} \wedge \underbrace{[\sqrt{x}=\sqrt{2}]}_{\substack{parametric\\data}} \underset{erroneous\ deduction}{\Rightarrow} \underbrace{[y=5]}_{solution} \tag{9}$$

This type of computation error is caused by a violation of logic processing, and is illustrated in the function range plot shown in Fig. 3.2b. In practice computation error can arise from incorrect arithmetic, software bugs, iteration, round-off, etc.

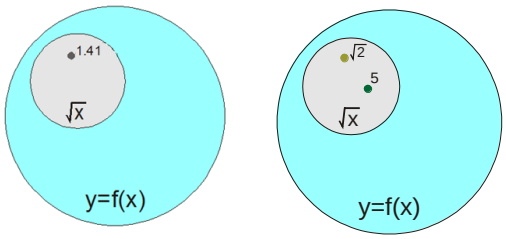


Figure 3.2 Computation error: a) approximation error; b) calculation error

In summary, the coherence test addresses computation error which can arise either in the numerical approximation of the mathematical model and its data (approximation error), or in the processing of the logic (calculation error).

4. Concepts of verification and validation in engineering simulation

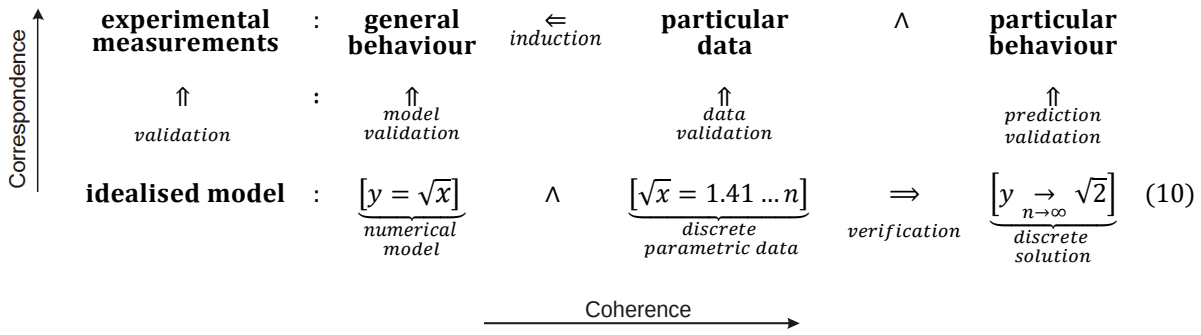
Up to this point, we have discussed syllogistic models and their relations to the scientific truth tests of coherence and correspondence. We will now apply these concepts to engineering simulation models and their associated verification and validation tests, and show that these tests are essentially the same as the scientific truth tests.

Widely accepted definitions of the verification and validation processes in engineering simulation are [1]:

verification*: the process of determining that a computational model accurately represents the underlying mathematical model and its solution.*
validation*: the process of determining the degree to which a model is an accurate representation of the real-world from the perspective of the intended uses of the model.*

As discussed in Section 2.1 and shown in Fig. 2, the coherence testing process is used to evaluate the computation error arising in both the approximation and calculation processes. As computation error is reduced, the discrete solution converges towards the analytical solution of the underlying mathematical model. Thus, according to the definition of verification given above, the verification test shown in Fig. 4 is essentially the coherence test shown in Fig. 2.

As discussed in Section 2.2 and shown in Fig. 2, the correspondence testing process is used to evaluate the idealisation error arising in the model error (major premises), parametric data error (minor premises) and predicted behaviour (conclusion). As the idealisation error is reduced, the idealised model should converge towards the real-world system that it describes (see also discussion in Section 7). Thus, according to the definition of validation given above, the validation test shown in Fig. 4 is essentially the correspondence test shown in Fig. 2.



(10)

Figure 4 The solution of the idealised model should correspond with increasing accuracy to that of the experimental measurements as the idealisation and computation error is reduced. In this figure, verification is characterised by convergence within rows, and validation by convergence within columns.

5. Engineering simulation example

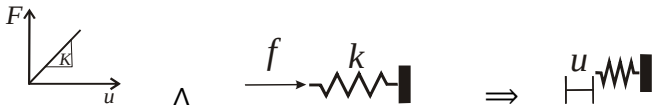
5.1 Simulation model

The following is an example of an application of the syllogism to a simple engineering simulation model of a single degree of freedom linear spring system (11) illustrated in Fig. 5.

This simulation model predicts the end displacement u, of a linear spring K, subject to an applied load F.

$$Ku - F = 0 \tag{11}$$

We write F/K to represent a combined stiffness and load parameter, f/k to represent the *value* of that parameter incorporating specific *data* (i.e. spring stiffness *value* k, and applied force *value* f), and $u = f/k$ to represent the solution of this data laden model. It can be seen from Fig. 5 that the idealised model is comprised of three basic components:



mathematical model F/K ∧ parametric data f/k ⟹ solution u

Figure 5 Basic components of an idealised model of a linear spring system, $Ku = F$

a) mathematical model F/K : represents the relationship between the main system variables;
b) parametric data f/k : represents the data laden model incorporating specific load and data values;
c) solution u : represents the displacement response of the model.

The relations between these three components will now be cast in syllogistic form as follows:

$$\underset{\substack{mathematical\\ model}}{[u = F/K]} \wedge \underset{\substack{parametric\\ data}}{[F/K = f/k]} \underset{deduction}{\Rightarrow} \underset{solution}{[u = f/k]} \tag{12}$$

which means that if $[u = F/K]$ and $[F/K = f/k]$ then $[u = f/k]$.

The first term in (12) is the major premise $[u = F/K]$, which represents the mathematical model. The second term is the minor premise $[F/K = f/k]$ which represents a data laden instance of that model (i.e. with particular values of material coefficients and loading included). Assuming that both premises $[u = F/K]$ and $[F/K = f/k]$ are true, logic necessarily entails the true conclusion $[u = f/k]$.

We employ approximation methods such as the finite element method to solve equations such as $[u = F/K]$, by establishing a set of approximate discrete equations $[F/K = f_h/k_h]$, where the approximation process simultaneously discretises both the model and the parametric data.

The numerical model converges to the mathematical model in the limit as the discretisation parameter h approaches zero, i.e. $[f_h/k_h] \underset{h\to 0}{\rightarrow} [f/k]$, and the approximate solution converges towards its analytical counterpart u as $h \to 0$ as shown in Fig. 6

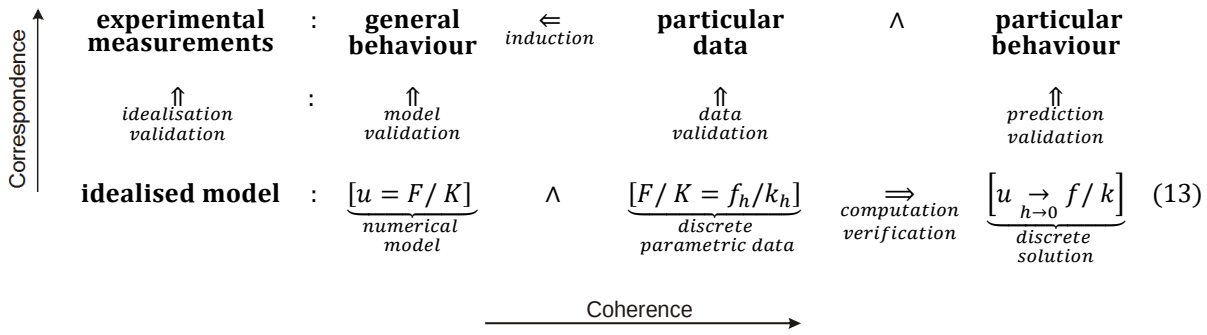


Figure 6 The solution of the idealised model should correspond with increasing accuracy to that of the corroborative system as the idealisation and computation error is reduced. In this figure, verification is characterised by convergence within rows, and validation by convergence within columns.

5.2 Simulation verification

In engineering simulation, the error sources to be verified arise within the computational modelling process as [3]:

a) approximation, consisting of:
    discretisation (e.g. spatial, temporal),
    truncation (e.g. eigenseries);
b) calculation, consisting of:
    arithmetical (e.g. mistakes, software bugs),
    iterative (e.g. divergence, round-off).

The verification process evaluates the differences between the computed and analytical solutions due to approximation error and calculation error, and is characterised by 'coherence'.

5.3 Simulation validation

In engineering simulation, the error sources to be validated arise within the idealised modelling process as [3]:

a) mathematical model, consisting of:
    governing equations (e.g. element or solution type),
    domain space (e.g. geometric extent),
    boundary conditions (e.g. fixed/free),
    constitutive relations (e.g. material model);
b) data, consisting of:
    loads (e.g. magnitude, direction, distribution),
    coefficients (e.g. material values).

The validation process evaluates the differences between the idealised model and the corroborative system (e.g. experimental results) due to mathematical model error and parametric data error, and is characterised by 'correspondence'.

6. A logical framework for simulation, verification and validation

This paper has shown that by casting a simulation model in syllogistic form, a logical taxonomy of simulation components and error sources naturally emerges (as summarised in Sections 5.2 and 5.3). It has also shown in Section 4 that the associated verification and validation processes are in fact forms of the scientific truth tests of coherence and correspondence; verification being characterised by coherence of the computational model, and validation being characterised by correspondence of the idealised model to experimental measurements. In summary, the two main error source categories emerging from this process are:

a) idealisation error;
consists of model and data error,
treated by the validation process (correspondence test),
convergence sought between the idealised model and the corroborative system,
validation challenges the behavioural accuracy of the idealisation,

b) computation error;
consists of approximation and calculation error,
treated by the verification process (coherence test),
convergence sought between the computational model and the idealised model,
verification challenges the numerical accuracy of the computation,
note: verification is a prerequisite for validation (see also Section 7).

The foregoing can now be consolidated and structured into a logical process of simulation model components and verification and validation activities (see Fig. 7), where the process sequence is:

1. A model of the real-world system is idealised and stated in mathematical terms.
2. This idealised model is numerically approximated to produce a computational model.
3. The discrete solution of this computational model is calculated.
4. The discrete solution is expanded to produce a continuous solution for comparison (verification process) against the analytical solution of the idealised model (see also comment b) in Section 7)
5. When acceptable solution convergence between the idealised and computational models has been achieved, the solution is interpreted in terms of the behaviour of the real-world system.
6. The predicted behaviour of the idealised model is compared (validation process) against the observed real-world behaviour (see also comments d) and e) in Section 7).
7. When acceptable convergence has been achieved between the behaviours of the idealised model and real-world system, the simulation can be deemed valid within that particular scope of testing.

The foregoing is summarised schematically as an overall process flow diagram in Fig. 7 [3].

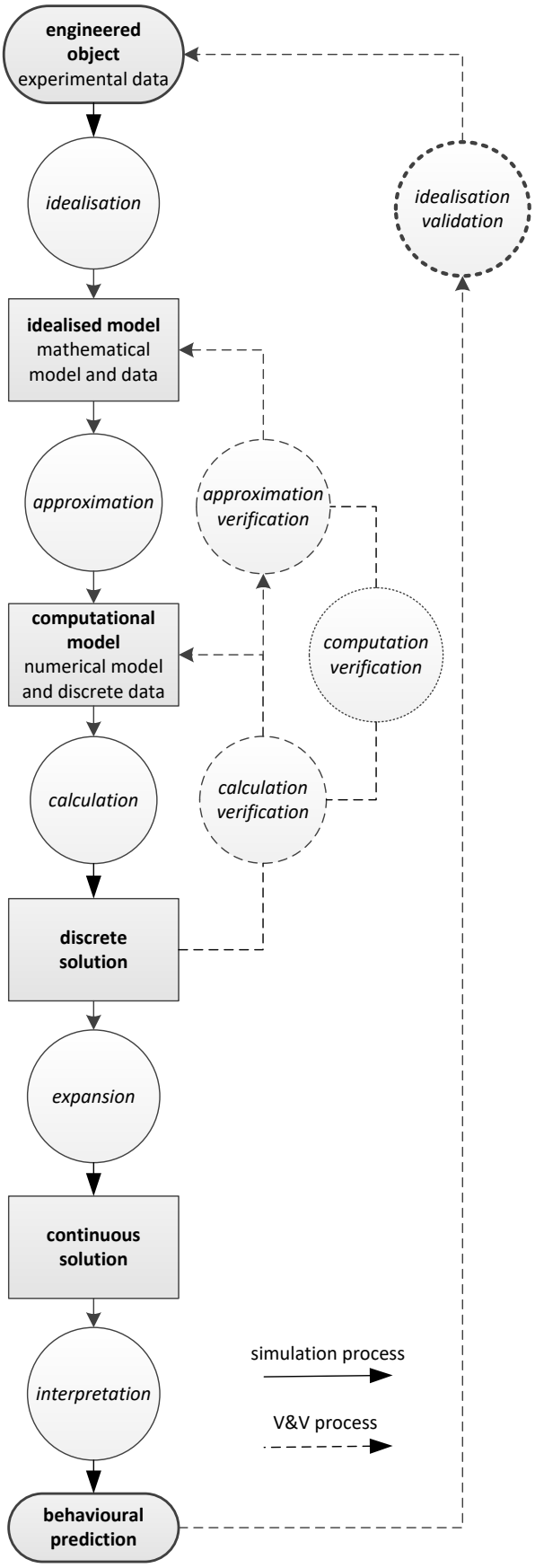


Figure 7 Overall simulation process flow diagram showing verification and validation activities

7. Discussion

It has been shown that the verification process tests the coherence of the computational model, and that the validation process tests the degree of correspondence between the idealised model and a corroborative system (e.g. experimental results).

Fig. 8 shows the structure of the error sources underlying the verification test of the computational model and the validation test of the idealised model.

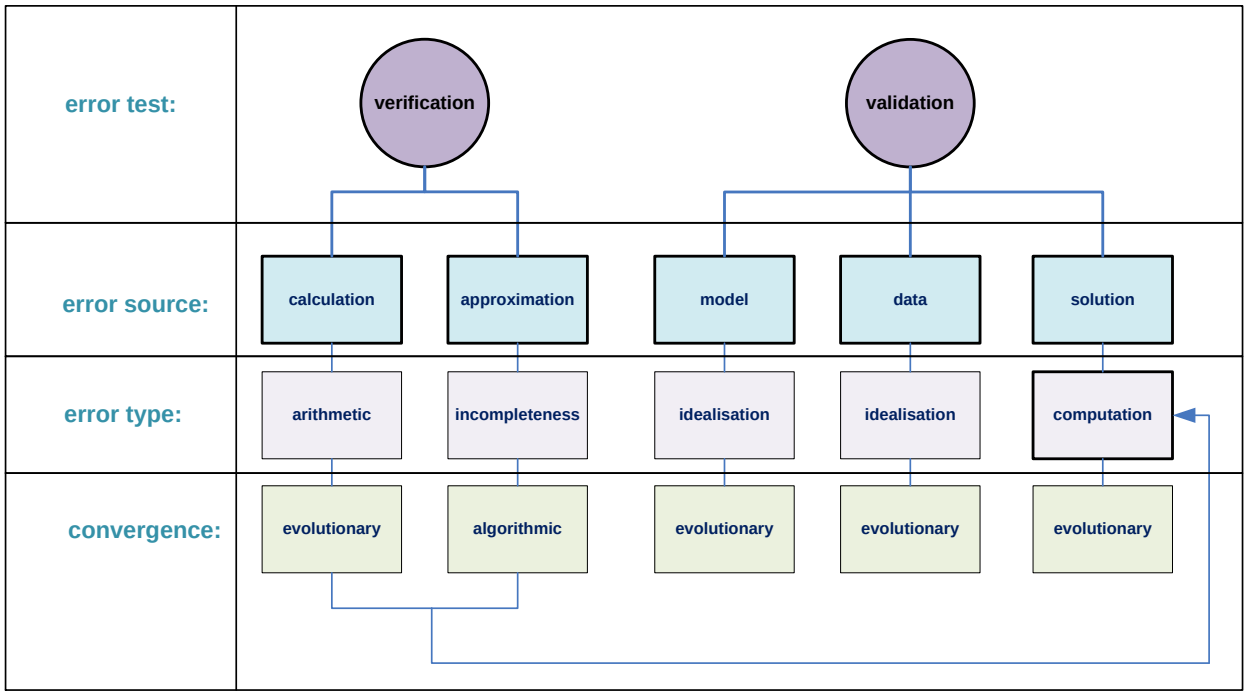


Figure 8 Error source framework showing components of, and relations between, verification (coherence of the computational model) and validation (correspondence between the idealised model and the corroborative system)

Comments

a) It is important to appreciate that convergence between the analytical solution and the computed solution is characteristically different to convergence between a real-world problem and an idealised model of it. This is because convergence between the computational and analytical solutions is governed by a predictable algorithmic process, whereas convergence between a real-world system and any model of it (see Fig. 8) is an evolutionary process. This is because no logical relation exists between the real-world and any description of it, and any convergence between the two entities evolves by an empirical, evolutionary process of random error correction.

b) As shown in Fig. 8, the computed solution constitutes an input to the model validation process since an analytical solution is not normally available. Therefore, the solution should not contain significant computation error, otherwise it will not be clear (for the purposes of error treatment) whether the error has arisen within the idealisation or computation processes, since the error in its

totality will be a complex combination from both sources. Therefore, successful model verification is a prerequisite for successful model validation.

c) While convergence of a sequence of approximate solutions (e.g. due to mesh enrichment) is algorithmic in nature and potentially predictable, convergence of calculation error (e.g. random bug fixing) is evolutionary and unpredictable (see Fig. 8). In practice however, assuming experienced analysts and mature software, the computational solution error is more likely to be dominated by numerical approximation characterised by algorithmic convergence. However, quantitatively assessing the degree of approximation error can be problematic since a variety of approximation error estimation methods may be applicable, each having a distinct theoretical basis and quantifying the error in a unique way.

d) Since the corroborative system needs to capture the relevant physics of the problem, an experimental corroborative system is preferred over experiential, analytical or simulation alternatives. However, even experimental models may fail to capture all the relevant physics, for example, due to the presence of unanticipated, and therefore unmeasured, loading and failure modes. Experimental procedure errors (e.g. metrological mistakes) and scientific reduction errors tend to occur randomly but diminish as the experimentation cycle matures.

e) Inputs to the model validation process will include a verified computational solution (ideally accompanied by numerical error estimates) together with the idealisation assumptions. In practice, the validation will typically involve a comparison of the idealised model's predicted values with those from an independent source such as experimentation and/or observation. However, the mathematical model and data assumptions inherent in the idealised model also need to be reviewed in order to assess their validity, independently of the accuracy of the computed solution.

f) Where model validation is unsuccessful, the sources of error in the model will need to be identified and treated, the model improved, and its predictions revised. This improvement cycle needs to be continued until an acceptable degree of validation is achieved, within accepted bounds of accuracy, risk and cost.

g) Software verification is an independent prerequisite for the simulation process, whereby software must be shown to work correctly before application, particularly with respect to the simulation type to which it will be applied [5]. However, some software bugs will inevitably find their way into the simulation process despite best efforts at prerequisite software verification, ultimately surfacing as calculation error. On identification of a software bug, the practical response would be to halt the simulation, fix the bug (or find a work-around), run the code verification process again, and then repeat the simulation. It is noted that the software verification process is identical to the computation verification process shown in Fig. 7, which begins with the idealised model (having an analytical solution) and ends with the continuous solution (for comparison with the analytical solution).

h) In this paper, only one major premise and one minor premise were used in the conceptual model template (from which an idealised model is created), but additional minor premises can also be added to limit the domain and range of the model (model constraints), e.g. $a < x \leq b$ or $y < d$.

References

1. Guide for Verification and Validation in Computational Solid Mechanics, ASME V&V 10-2006.

2. Russell , B., The Problems of Philosophy (Ch. XII), 1912.

3. Smith, J. M., Quality Management in Engineering Simulation, NAFEMS, 2014.
http://www.nafems.org/publications/browse_buy/browse_by_topic/qa/nafems_qss_primer_2014_-_r0116/

4. Okasha, S., Philosophy of Science: A Very Short Introduction (pp. 40-52), Oxford University Press, 2002.

5. Smith, J. M., ed., NAFEMS QSS : Engineering simulation – Quality management systems – Requirements, NAFEMS, 2014.
http://www.nafems.org/publications/browse_buy/browse_by_topic/qa/nafems_qss_2008_/