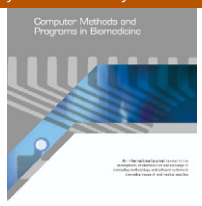
COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE 102 (2011) 295–304

journal homepage: www.intl.elsevierhealth.com/journals/cmpb



Graphical simulation environments for modelling and simulation of integrative physiology

Violeta Mangourova[a,*], *John Ringwood*[a], *Bruce Van Vliet*[b]

[a] *Dept. of Electronic Engineering, NUI Maynooth, Ireland*
[b] *Biomedical Sciences Division, Memorial University, St. John's, Newfoundland, Canada*

ARTICLE INFO

Article history:
Received 27 April 2009
Received in revised form
17 April 2010
Accepted 8 May 2010

Physiology
Mathematical model
Computer model
Simulation

ABSTRACT

Guyton's original integrative physiology model was a milestone in integrative physiology, combining significant physiological knowledge with an engineering perspective to develop a computational diagrammatic model. It is still used in research and teaching, with a small number of variants on the model also in circulation. However, though new research has added significantly to the knowledge represented by Guyton's model, and significant advances have been made in computing and simulation software, an accepted common platform to integrate this new knowledge has not emerged. This paper discusses the issues in the selection of a suitable platform, together with a number of current possibilities, and suggests a graphical computing environment for modelling and simulation. By way of example, a validated version of Guyton's 1992 model, implemented in the ubiquitous Simulink environment, is presented which provides a hierarchical representation amenable to extension and suitable for teaching and research uses. It is designed to appeal to the biomedical engineer and physiologist alike.

1. Introduction

At the time of its conception (1972), the integrated physiology model developed by Guyton [1] was a breakthrough in many respects:

(1) It assembled the available knowledge on the dynamics of the body's circulatory components and how they interacted with each other,
(2) It presented a diagrammatic form, which allowed the totality of the model to be viewed and interactions examined, and
(3) The model was specified using the basic components of integrators, summers and (sometimes nonlinear) gains, the fundamental building blocks of analogue computers, facilitating computation.

Clearly, the longevity of this model is testament to its quality and usefulness. Unfortunately, developments in integrative physiological modelling have not kept pace with further discoveries in physiological science and developments in computing. In particular, the absence of an agreed computing platform, which can serve current teaching and research purposes, *and* be added to and updated by the physiology community at large, is disappointing. Recently, a number of large-scale initiatives have emerged, such as the IUPS (International Union of Physiological Sciences) Physiome Project [2], which attempt to directly address this issue. However, there is the issue of the universal adoption of a single standard in a situation where a number of competing standards exist and the associated question as to whether a single computing platform can cater for needs at all levels of detail and timescale. Furthermore, in the multi-disciplinary world of physiological modelling, where mathematicians, engineers, scientists

* *Corresponding author*. Tel.: +353 879886557.
E-mail address: violeta.i.mangourova@nuim.ie (V. Mangourova).
0169-2607/$ – see front matter
doi:10.1016/j.cmpb.2010.05.001

and medical practitioners collaborate, researchers may favour modelling and computing tools which are prevalent in their own original discipline area.

This paper looks at the modelling and simulation requirements of integrative physiology and the range of tools available, and under development, which address this issue. We propose a graphical modelling and simulation environment as an integrated solution and present a validated Simulink version of Guyton's most recent (1992) model [3] as an example of such a solution, together with a discussion of outstanding issues which need to be resolved.

2. Physiology modelling and simulation

Traditionally, mathematical models are presented through a series of equations (algebraic and/or differential), with the possibility of a block diagram representing the interconnections of various subsystems. Guyton's model represents such a block diagram, where the individual building blocks are described by particularly simple elements: summers, integrators and (possibly nonlinear) gains. This elementary form was particularly chosen to suit implementation on an analogue computer. Various digital computing models have derived from Guyton's model, with a variety of characteristics. The Fortran implementation of Guyton's model simulated using MODSIM [4], and based on a difference equation form relying on forward-difference discretization, is used as a teaching and research tool [3]. The sequential command form of the code makes the wider interactions in the model difficult to visualise. Coleman's extension of Guyton's work [5] via the QHP (Quantitative Human Physiology) model was available only in executable form (for teaching purposes) until recently, when an eXtensible Markup Language (XML) version of the model was released [6,7]. This extended model, however, is difficult to read and visualise, since no diagrammatic form exists.

2.1. *Modelling*

In order to take a structured approach to the discussion of modelling and simulation for integrative physiology, one must question the objective of the modelling and simulation exercises. In modelling, the following set of objectives are pertinent:

- Develop a set of continuous-time equations which represent the underlying physiological components [8–11],
- Parameterise the equations on experimental data [12], and
- Validate the model on previously unseen data [13].

Regarding the mapping of the physiology to a set of equations, a variety of levels of granularity and transparency are possible:

- In relation to transparency, models have the following forms:
  - White-box, where each parameter represents a physical physiological quantity,
  - Grey-box (and the sub-classes of off-white, smoke-grey, steel-grey and slate-grey [14]), with various levels of connection to the underlying physiological structure, and
  - Black-box, where the model simply reproduces the experimental output data, given the same stimulus, but the internal model structure bears no resemblance to the physiology.
- A variety of timescales may be encapsulated by a single, integrated model, and
- A model may be hierarchically built using progressively larger 'component' sizes. See, for example, the hierarchy suggested by Hunter and Nielsen [15] in Fig. 1.

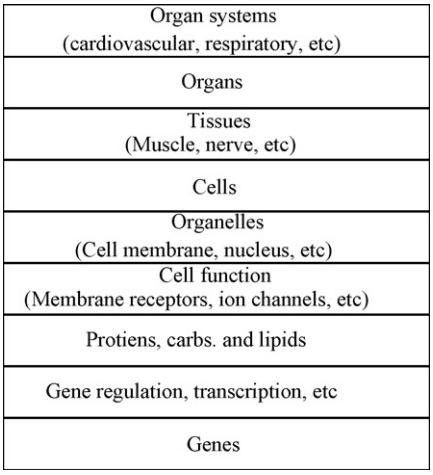


Fig. 1 – Multiscale modelling hierarchy [15].

However, it is reasonably clear that a digital computer implementation should not necessarily have to go through the steps required for simulation on analogue computers, but rather should focus on representations that relate to physical or intuitive quantities and representational levels that assist visualisation and ease of use.

A number of formalised modelling methods [16], some of which have the ability to deal with non-homogeneous systems (i.e. systems with mixtures of mechanical, flow, and electrical elements), e.g. bond graphs [17,18], are available, which systematically take the modeller through Step 1 described in Section 2.2. A variety of system identification routines [12] can be used to estimate parameter values from experimental data.

2.2. *Simulation*

Physiological modelling gives us a representation of a real system, parameterised either in terms of physiological quantities which are known (constants and variables) or a black-box model that simply has the capability to produce an 'output' signal in response to a stimulus. In order for the representation to simulate the system, it needs to be implemented on a computer. Some model forms (e.g. black-box models), which are built using computers, are easily directly simulated, while others require inputting to a simulation engine. One can document the steps required in simulating a system (physiological or otherwise) on an *analogue* computer as:

(1) Model equations are assembled—structure and form, including interconnections,
(2) Model parameters are determined (white, various shades of grey, black),
(3) Model equations are then decomposed into fundamental components of:

- Summation nodes,
- Integrators (the fundamental dynamic element in an analogue computer), and
- Gains, including nonlinear gain elements.

For simulation on a digital computer, step 3 above is replaced by:

(4) Model equations are discretized, and
(5) Equations are implemented in computer code.

It is clear that the parameterisation of Guyton's model was motivated by analogue computer simulation and, while their use is now virtually redundant, they do not require the extra step of choosing a discretization method, as required in a digital computer simulation. At this stage, it is useful to observe that there is no 'perfect' form of discretization which preserves all the properties of the original continuous-time system (e.g. time response, frequency response, etc.) with exact fidelity [19].

Therefore, a model simulated using different discretization techniques may produce significantly different responses, depending on the nature of the system. Certain nonlinear systems, stiff systems (systems containing both fast and slow dynamics) and systems containing algebraic loops may all be sensitive to the simulation engine and/or discretization used. Most discretization methods are based on numerical integration techniques (e.g. Euler, Runge-Kutta, methods based on the Adams-Moulton and Adams-Bashford families, etc.) [20–22], with some based on frequency response mapping (e.g. pole-zero mapping) [23]. For example, a Fortran implementation of Guyton's model [3] uses a (first order) forward-difference discretization method. Other methods produce authentic responses for particular inputs, such as impulses, steps, etc. [24].

Therefore, we need to decide what we want to achieve in simulation [25] and the ability to reproduce the output of a published model [26] may require more than the ability to read the set of continuous-time equations using a common computer-interpretable standard. One can include the equations, so long as the simulation metadata is also included, i.e. discretization method, word-length, how to deal with algebraic loops, etc.

3. Modelling and simulation environments

Currently, a wide variety of modelling and simulation platforms exist, many of which suggest to offer the user customisation and facilities specific to certain application domains. We briefly review the classes of such platforms in the remainder of this section, with the focus on their possible use for integrative physiology.

3.1. *Diagrams and equations*

The original (for example, see [27]), and one of the best environments for the communication of mathematical models is using block diagrams and differential or difference equations. Specification of sets of continuous-time equations (with associated block diagrams showing subsystem interconnection) still dominates the published literature. This form, usually specified in continuous-time, utilises well-accepted standard mathematical and block diagram notation, but is not directly computer readable and is purely for representation purposes, with no simulation potential or information.

3.2. *Markup languages*

Markup languages have become popular over the past two decades in attempting to provide universal specification of content, for example the HyperText Markup Language (HTML) in the specification of presentation for web browsers and UML (the Unified/Universal Modelling Language), which is a standardised visual specification language for object modelling in software engineering. More generally, XML (eXtensible Markup Language) was developed as a framework within which other 'MLs' could be generated for specific purposes, an example of which includes MathML [28], which is used for the specification of mathematical expressions and can form the basis for mathematical model description. MathML has found currency in the specification of econometric models, for example in the Journal of Econometrics.

The main advantages of the markup languages is that they reduce relatively complex content to very simple, but verbose, ascii form. This guarantees maximum transportability and an independence from machine-specific implementations. It also provides some element of 'future-proofing', in that the way in which the ML content is utilised is not specified exactly, so that future technology can both produce the ML content efficiently and 'utilise' it effectively. A number of domain-specific markup languages have evolved, including:

- The set of markup languages associated with the IUPS Physiome project (see Fig. 2, and also [29]), namely [30]:
  - *ProteinML* which can describe pathway models,
  - *CellML* [31], which provides a representation of the mathematical relationships of biophysical models at the cell level,
  - *TissueML*, which describes models at the tissue level,
  - *FieldML* [32], describing spatially and temporally varying fields (such as electric fields) related to cell structure,
  - *AnatML*, which deals with physiology at the organ level, and
  - *PhysioML* [33], which addresses the organ system and organism level.
- CML (Chemical ML) [34], which provides a representation for managing molecular information, from macromolecular sequences to inorganic molecules and quantum chemistry.
- SBML (Systems Biology ML) [35], which describes models of biochemical networks, such as signal transduction, metabolic pathways and gene regulation, and
- SysML (Systems ML) [36], which caters for 'systems' engineering, typically supporting block diagrams of interconnected systems.

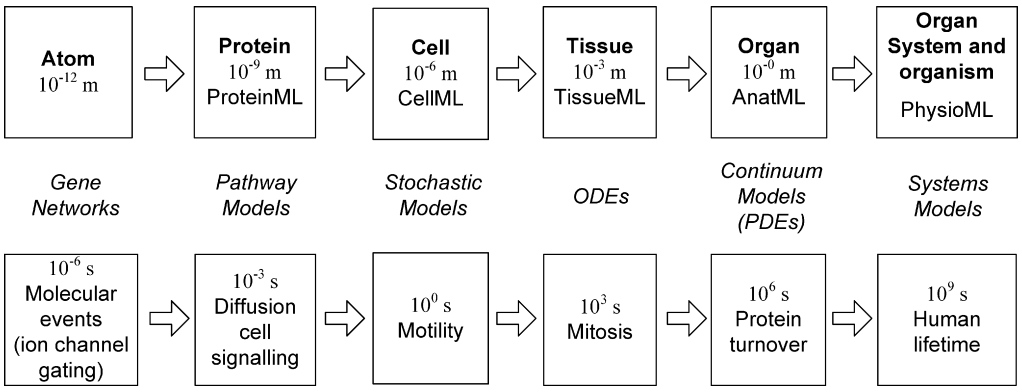


Fig. 2 – Spatial (top) and temporal (bottom) scales encompassed by the Human Physiome Project along with corresponding markup language (based on [30]).

CellML, promoted by the IUPS Physiome Project [2], is a subset of Content MathML, which is the 'active' version of MathML (the other one is Presentation MathML). CellML has been suggested as a potential solution platform for cell function and to standardise the interface to other computer programs and probably represents the most developed of the IUPS Physiome 'MLs'. A CellML model contains both the essential model equations, written in ascii/markup, and associated 'metadata' which contains [31]:

- The units in which the variables associated with the model component are measured,
- Author, literature reference, creation date, and
- Biological context.

An element of hierarchy is also provided for in CellML via the *encapsulation* and *containment* grouping relationships. Simulation tools are available for CellML models, via a CellML application programming interface (API) [31], and graphical tools for the development of CellML models are under development. CML and SBML appear to overlap in scope with some of the IUPS Physiome 'MLs', while PhysioML [33] is somewhat unique in its ability to control the interface display and can perform 'computational steering', which refers to the selection of particular variables which can be altered during computation. However, there is currently no means to incorporate model descriptions in PhysioML though there may be an option in the future to achieve this using MathML. SysML operates at a high (systems) level and customises UML for systems engineering applications. It allows for the incorporation of 'systems-level' information, including hardware, software, information, processes, personnel, and facilities. Virtually all of the 'MLs' (including supporting software, which can include modelling and simulation tools) are well documented on individual websites, which are easily accessed using popular search engines.

3.3. *Graphical modelling and simulation environments*

There is a wide range of graphical modelling and simulation environments available. In general, they integrate the functions of modelling and simulation, where models are built up in block diagram form and a simulation engine (with selectable parameters) can be used to run the model.

The range of graphical modelling/simulation environments includes the following: Modelica/Dymola (The Modelica Association), Scicos (The Scilab Consortium), Simulink (The Mathworks Inc.), ExtendSim/Extend 6 (Imagine That Inc.) , 20-sim (Controllab Products B.V.), Easy5 (Mechanical Dynamics Inc.), MATRIXx (National Instruments Corporation) and VisSim (Visual Solutions Inc.). These are all proprietary software packages, apart from Scicos, while the open-source Modelica [37] modelling environment requires the commercial Dymola application for simulation. In general, graphical simulation packages offer very fast development times for models, assisted by libraries of standard blocks and, in many cases (e.g. Simulink/MATLAB (The Mathworks Inc.), Scicos/Scilab), an associated macro language parser which allows high-level calculations. Of the listed packages, Simulink [38] is probably the most popular, though it has become relatively expensive and it is unclear whether commercial motives or necessary technical changes have resulted in compatibility difficulties between the many releases, which seem to come at an ever increasing rate. The Scilab/Scicos [39] environment is unlikely to suffer the same fate, being an open-source platform, but does not yet offer the same level of functionality as MATLAB/Simulink. Modelica, which is promoted by the non-profit Modelica Association, has the advantage of being object oriented and can model complex heterogeneous systems. The most popular simulation engine for Modelica, Dymola is, however, a commercial product, though other (free) simulation engines, including OpenModelica and Modelicac are also available. In addition, a Modelica to XML translation is available [40], which would allow to share the model in a generic format.

In terms of capability, Simulink performs best with pure ordinary differential equations (ODEs), with algebraic equations solved iteratively, which can require excessive computation (though insertion of small artificial delays can circumvent this). Modelica/Dymola provides a more natural way to simulate differential algebraic equations (DAEs) directly. In terms of model storage, Simulink uses a proprietary (and quite opaque) format, though translators [41,42] are available to convert between representations, but these are unlikely to preserve the full detail of the original model description.

A particularly promising open-source offering under development is the ProMoT (Max Planck Institute) (Process

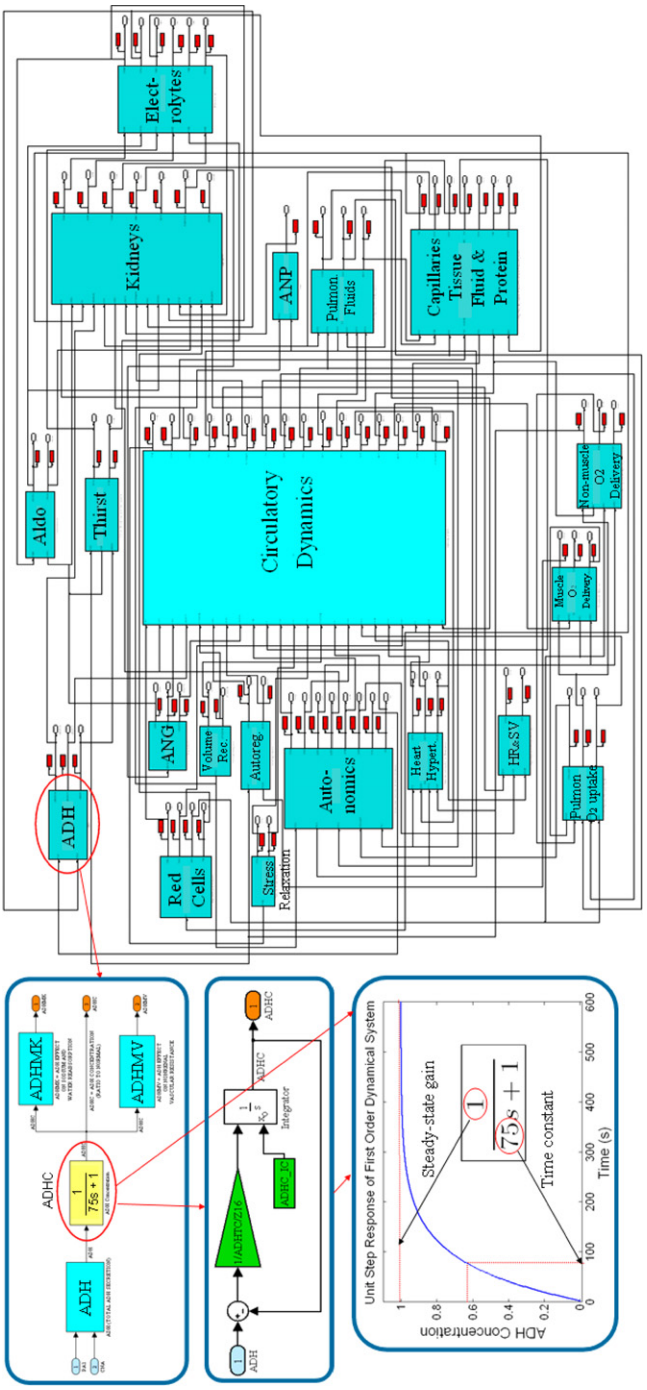


Fig. 3 – Our Simulink implementation of Guyton's 1992 model.

Modelling Tool) environment [43], which can build structured dynamic simulation models that are based on differential and algebraic equations. Developed originally for process engineering, it has been more recently applied to systems biology, with library modules available for application domains such as separation processes, membrane processes, fuel cells, metabolic cellular processes and signal transduction. Simulation is via the DIVA package, which uses some commercial libraries for numerical computation, but a further open-source simulation module (DIANA, Dynamic sImulation And Numerical Analysis tool) is currently under development. ProMoT uses symbolic reduction for efficient simulation, which deals effectively with algebraic loops. The scripting language, Python, is used as a command-line interface, and an interface is provided to MATLAB, with other interfaces possible using the SBML standard. Example applications which use ProMoT include [44,45].

Other graphical modelling/simulation tools, which have been developed for specific biomedical applications, include Cytoscape [46], which is an open-source software project for integrating biomolecular interaction networks with high-throughput expression data and other molecular states into a unified conceptual framework. The essential building block of Cytoscape is the network graph. Finally, SAAM II (University of Washington) [47] is a software environment having a command-line and graphical interface for tracer and pharmokinetic studies.

3.4. *High-level languages*

Following the development of the digital computer, continuous-time models could be implemented using high-level programming languages. Following the digital simulation process outlined in Section 2.2, this requires the user to 'manually' discretize the system and take care of algebraic loops leading to some individuality in the simulation results. The resulting high-level code is strictly sequential (with, of course, provision for looping and branching), making it rather difficult to interpret from a 'systems' perspective. The exception to such sequential processing is the possibility for parallel computation. However, the possibility of obtaining an exact custom map of physiology structure onto parallel computing hardware would be difficult to achieve and is not likely to be a cost-effective solution, nor would it provide significant extra accuracy or intuitive appeal.

High-level languages which have been employed for dynamic simulation include Basic, Fortan, Pascal, Ada, C, C++, Java, etc. The main advantage of customised coding using a high-level language is the speed gain over, say, graphical simulation environments. This speed advantage has resulted in the continued popularity for certain computationally intensive application areas, e.g. marine hydrodynamics [48].

4. Guyton's model in Simulink

When modelling of physiological systems is concerned, the question of what level to define the model at, always arises. Models can be defined at a multitude of levels and potentially models at different spatial levels and different levels of timescale resolution can be integrated. The necessity of such integrated implementation needs to be assessed taking into consideration the computational intensity, and purpose of the model, and the aptness of carrying a large complexity if it is not necessary.

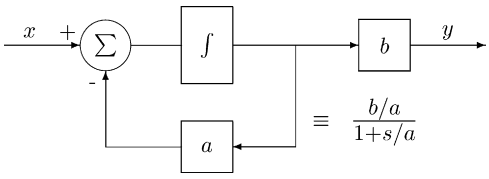


Fig. 4 – First order system with steady-state gain b/a and time constant $1/a$.

An example of model implementation at the 'macro' (computational physiology [49], systems physiology, organ systems) level has been shown by Cabrera [50], who looks at the dynamics of lactate production during exercise. The model is in a block diagram form and at the 'Organ systems' (or inter-organ) level in Fig. 1.

Guyton's model implemented in this study in the Simulink environment is at the 'Organ systems' level also and an example of the systems and subsystem levels is shown in Fig. 3. The top level is shown and two subsystem levels, including one at the lowest first order dynamic level.

The 1972 published version of Guyton's model has been implemented by a number of researchers ([51,52]) in the Simulink format suggested in this study. A 1986 version of Guyton's model, which was not published, has also been implemented by Kofranek et al. [51].

In this present study, the most current version of Guyton's model from 1992 was implemented in Simulink, using both model diagrams, similar to those published in 1972, and Fortran code implementation of the model equations in discrete-time [3]. Our Simulink version of the model was implemented in continuous-time, i.e. using continuous-time integrator blocks, while the discretization method for model simulation can be chosen from a variety of solvers, with either a fixed or variable step size. Since the model size, and number of variables and parameters, has increased immensely since the first version of the model from 1972, multiple layers of subsystems were used. At the lowest level, a first order dynamic system was implemented as in the original model diagram [1] with an integrator, a gain and a summer, as in Fig. 4, and was subsequently 'masked' as a transfer function in the s-domain in the form of:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{k_{ss}}{1 + s\tau}. \quad (1)$$

This representational form allows an easy deduction of the system's steady-state gain $k_{ss} = b/a$ and time constant $\tau = 1/a$, and is much preferred to the original lower level gain-summer-integrator implementation. Fig. 5 shows a sample step response of such a first order system, where $k_{ss} > 1$.

Some issues transpired during the model implementation and validation. In the Fortran implementation of the model [3], first order 'damping dynamics' were included. These seem to have been added to avoid numerical instability; however

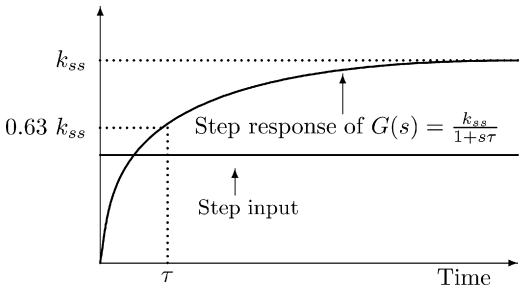


Fig. 5 – First order system step response with steady-state gain $k_{ss} = b/a > 1$ and time constant $\tau = 1/a$.

it is not completely clear whether these dynamics have any physiological function or are simply a feature of the model implementation and, in particular, the possibly unsuitable discretization method. In addition, most of these discrete-time first order dynamic 'damping' equations included specific integration step sizes, as expected, while a few of the equations did not feature these step sizes. A discrete dynamic equation without an integration step size has no continuous-time counterpart and cannot be implemented in our Simulink model without making unreasonable assumptions, such as that the step size is included in the gain parameter, which defines the time constant of the system.

An example of a first order damping equation, the calculation of angiotensin secretion, where a step size is included, is as follows:

$$\mathrm{ANX1} = \mathrm{ANX1} + \frac{\mathrm{ANX} - \mathrm{ANX1}}{\mathrm{ANV} * I} \tag{2}$$

where ANX refers to the angiotensin secretion, ANX1, actual ANX after damping, ANV, time constant of angiotensin secretion, and I, the time incrementation step.

Alternatively, an example of a dynamic equation without the necessary incrementation step size is the damping of renal peritubular capillary reabsorption (RFAB1):

$$\mathrm{RFAB} = \mathrm{RFAB} + \frac{\mathrm{RFAB1} - \mathrm{RFAB}}{\mathrm{RFABDP}} \tag{3}$$

where RFAB refers to RFAB1 after damping, RFAB1, renal peritubular capillary reabsorption, and RFABDP, RFAB damping factor.

We can only assume that the step size was incorporated into the RFABDP time constant parameter. This is a very poor implementation solution, since the integration step sizes seem to vary during model simulation, giving somewhat unpredictable results.

Our Simulink model was validated against the 1992 version of Guyton's model, implemented in Fortran and simulated in the MODSIM environment [3]. A sample 'reference' experiment, similar to those described in [1,52], was simulated by reducing kidney mass to 30% of normal and increasing salt intake to 9 times normal at the start of the experiment. The responses of the MODSIM implementation and our Simulink version of the model are shown in Fig. 6. It is seen that the Simulink model output is comparable to the MODSIM implementation, while the small difference in response could be due to a number of factors, namely the different discretization methods used and the issues with damping dynamics, as explained earlier in the paper. A variable step ode15s

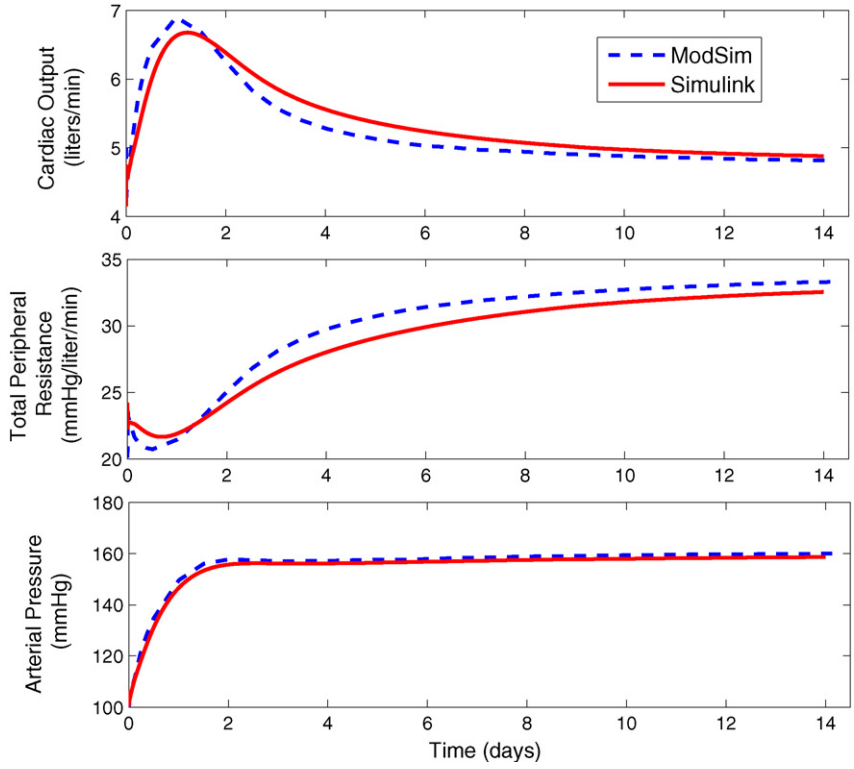


Fig. 6 – Comparison between MODSIM and Simulink implementations of Guyton's 1992 model for an experiment with reduced kidney mass and increased salt intake.

(stiff/NDF) solver was used for the Simulink model, as the system is highly stiff, with time constants varying from a few seconds to over thirty days. It is possible to use a fixed step integration method, however due to the stiffness of the system simulation can take a very long time to complete, since the step size must be chosen according to the shortest time constant.

The Simulink models by Kofranek et al. [51] and especially the 1989 version, also produce similar results to ours. The model presented in this work, however, is more recent and also presents the first order dynamics in transfer function form, which is more intuitive and gives direct information on steady-state gain and time constant.

The Simulink implementation of Guyton's model under the SAPHIR project [52] differs significantly from the model presented here. Thomas et al. [52] specified the model in discrete-time, using discrete-time integration blocks and a forward Euler method. In addition, all subsystems were treated as 'atomic subunits', thus all computations within a subsystem were executed before moving to the next subsystem. Variable integration step sizes were also employed. All of these described features mimic closely the Fortran implementation of the model and thus do not provide the desirable properties of:

- Providing a generic representation of the underlying continuous-time physiology, and
- Using a numerical integration technique, appropriate for the stiff system under consideration, which can be applied consistently across all model subsystems, without recourse to any custom discretization within the model.

Correctly addressing the above two issues is vital if a transportable, open-access model is to be developed. Nevertheless, the use of custom discretization can, in some cases, result in shorter simulation times, but also results in a model where the model and discretization are inextricably intertwined.

5. Discussion

The evolution strategy of a state-of-the-art model should be managed in a 'community' way, where the essential objective of any evolution of the modelling standard is to preserve the original inter-operability ethos, while taking advantage of technological developments and addressing perceived weakness in the 'standard' as experience with it builds. Ironically, MATLAB and its Simulink toolbox were built on the original EISPACK [53] and LINPACK [54] open-source libraries. EISPACK AND LINPACK, developed as Fortran libraries to implement eigensystem and linear algebra routines and originating from the Argonne National Laboratory, aimed to be portable, robust and reliable. The Numerical Algorithms Group (NAG) then provided these routines in a commercial package (somewhat similarly to the way in which Red Hat offer Linux). Currently MATLAB/Simulink is a commercial product with associated features and characteristics resulting from predominantly commercial objectives. New versions do not have backward compatibility and forward compatibility is not ensured either. On the other hand, open-source standards generally preserve inter-version compatibility as a priority, since they aim to serve community, rather than commercial, needs.

The benefit of using a well developed commercial tool such as MATLAB's Simulink lies in the fast and easy model development and validation. Our Simulink model was developed in a single week, though requiring significantly more time to 'validate' it against the 1992 MODSIM Fortran implementation (mainly due to the custom implementation features in the Fortran code). This highlights two issues:

(1) The model development time in Simulink can be extremely fast, once the underlying physiological structure is known, even when the model is complex, and
(2) Discretization issues can be significant in getting agreement between different implementations of the same model.

Due to the large number of modelling/simulation environments, it may be difficult to impose a single standard on all researchers, who may have very different backgrounds. Simply using a common interface could be a practical solution to the problem (e.g. CellML or QHP); however metadata containing information about the numerical integration method needs to be included.

In summary, three main components are necessary for successful model development, implementation and improvement:

- A modelling tool—preferably graphical, which allows networks or interconnections of subsystems to be built up in a hierarchical manner,
- Simulation tools, with the ability to simulate ODEs, DAEs and PDEs and implement a variety of numerical integration methods, including the efficient solution of stiff systems, and
- A non-proprietary interface standard, probably XML-based, which allows models to be readable across a wide variety of platforms.

6. Conclusions

Over thirty years ago, Dr. Arthur Guyton suggested that circulatory physiology was starting to change into and use the techniques of engineering science [1] and his model made large strides towards this change. Unfortunately, his famous circulatory physiology model has not developed significantly since 1992 in the practical form of a block diagram of systems and subsystems. A variety of simulation environments have been examined in this paper with a view to the mathematical modelling of large-scale physiological systems; however not many environments allow the desired visual diagrammatic representation and speed of implementation.

Some computer implementations of the original 1972 Guyton model exist [52,51], as well as an implementation of a 1986 version of the model [51]; however the Simulink model presented in this paper is the first implementation of the most recent 1992 version of the model. If a core model is to be created and made available for extension/improvement, it should

be the most up to date model recognised by the research and teaching community.

Guyton's model has been expanded previously under the QHP project, and with the QHP computer program now being open source, the community can, in theory, contribute to it. This new model, however, is implemented in an XML format and is difficult to read. In addition, appropriate structures have not been put in place to deal with model additions and a repository system for alternative model subsystems has not been created.

Technical implementation issues also make QHP unsuitable for development as a community project. Differential equations are implemented using one of three provided methods, where the model developer has to nominate a particular discretization method for each equation. The discretization methods include a first order (forward-difference) Euler method and a stiff system discretization method, while the third method is undefined. The QHP model is defined in an XML format, while the model solver can be downloaded only in executable form from the QHP website, but details of its features are not available. QHP could be a good solution with further development. A graphical environment for faster development time would be very beneficial while better transparency and documentation in relation to the solver is required. Finally, a repository system, catering for community contributions to the model needs to be created, if it is truly to be treated as a open-source resource.

For the present, however, Simulink remains one of the few viable options for large-scale model visualisation and implementation. Both development and validation are fast and easily executed with an intuitive graphical environment and results display. MATLAB also allows us to emulate the continuous-time nature of physiological systems by using continuous-time dynamic blocks and to select an appropriate solver for the system. Nevertheless, compatibility issues impinge significantly on the utility of Simulink as a community platform on which an integrated physiology model can evolve.

Finally, if a common XML standard is to be agreed upon, as a model sharing interchange standard, three important features must be complied with:

- The model needs to be specified in continuous-time as a fundamental representation,
- Discretization information needs to be included as metadata, and
- Graphical editing tools, allowing block diagram representation with hierarchical visualization capabilities must be available.

These features would ensure the successful reproduction and simulation of the model in any chosen environment, given suitable conversion tools, such as in [40–42].

Conflict of interest statement

None declared.

Acknowledgements

The authors would like to acknowledge research funding provided by the Irish Research Council for Science Engineering and Technology and NUI Maynooth, under the John and Pat Hume Scholarships programme. The authors would also like to thank Jean-Pierre Montani for providing the diagrams and Fortran code of the 1992 version of Guyton's model.

REFERENCES

[1] A. Guyton, T. Coleman, H. Granger, Circulation: overall regulation, Annu. Rev. Physiol. 34 (1972) 13–44.
[2] P. Hunter, E. Crampin, P. Nielsen, Bioinformatics, multiscale modelling and the IUPS Physiome Project, Brief. Bioinform. 9 (4) (2008) 333–343.
[3] J.-P. Montani, Personal communication.
[4] J.-P. Montani, T. Adair, R. Summers, T. Coleman, A. Guyton, A simulation support system for solving large physiological models on microcomputers, Int. J. Biomed. Comput. 24 (1989) 41–54.
[5] T. Coleman, J. Randall, HUMAN: a comprehensive physiological model, Physiol. Teach. 31 (1) (1983) 15–21.
[6] S. Abram, B. Hodnett, R. Summers, T. Coleman, R. Hester, Quantitative circulatory physiology: an integrative mathematical model of human physiology for medical education, Adv. Physiol. Educ. 31 (2007) 202–210.
[7] R.L. Hester, T. Coleman, R. Summers, A multilevel open source integrative model of human physiology, FASEB J. 22 (2008) 756–758.
[8] A. Garfinkel, A mathematics for physiology, Am. J. Physiol. Regul. Integr. Comp. Physiol. 245 (1983) R455–R466.
[9] J. Keener, J. Sneyd, L. Sirovich, S. Wiggins, L. Kadanoff, J. Marsden, Mathematical Physiology, Springer Verlag, 1998.
[10] E. Carson, C. Cobelli, Modelling Methodology for Physiology and Medicine, Academic Press, 2000.
[11] J. Ottesen, M. Olufsen, J. Larsen, Applied Mathematical Models in Human Physiology, SIAM, 2004.
[12] L. Ljung, System Identification: Theory for the User, 2nd ed., Prentice-Hall, 1999.
[13] C. Cobelli, E.R. Carson, L. Finkelstein, M.S. Leaning, Validation of simple and complex models in physiology and medicine, Am. J. Physiol. Regul. Integr. Comp. Physiol. 246 (2) (1984) R259–R266.
[14] L. Ljung, Perspectives on system identification, in: M. Chung, P. Misra (Eds.), Milestone Reports and Selected Survey Papers, Seoul, 2008, pp. 47–59.
[15] P. Hunter, P. Nielsen, A strategy for integrative computational physiology, Physiology 20 (2005) 316–325.
[16] L. Ljung, T. Glad, Modeling of Dynamic Systems, Prentice-Hall, 1994.
[17] P. Gawthrop, G. Bevan, Bond-graph modeling, IEEE Control Syst. Mag. 27 (2) (2007) 24–45.
[18] V. Le Rolle, A. Hernandez, P. Richard, J. Buisson, G. Carrault, A bond graph model of the cardiovascular system, Acta Biotheor. 53 (2005) 295–312.
[19] U. Ascher, L. Petzold, Computer Methods For Ordinary Differential Equations And Differential-algebraic Equations, SIAM, 1998.
[20] E. Hairer, S. Nørsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, Springer Verlag, 1993.
[21] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems, Springer Verlag, 1996.

[22] C.V.L.C. Molar, Nineteen dubious ways to compute the exponential of a matrix, twenty five years later, SIAM Rev. 45 (2003) 3–49.
[23] N. Hori, R.J. Cormier, K. Kanai, Matched pole-zero discrete-time models, IEE Proc. Control Theor. Appl. (Part D) 139 (3) (1992) 273–278.
[24] N. Hori, A. Mori, P. Nikiforuk, A new perspective for discrete-time models of a continuous-time system, IEEE Trans. Autom. Control 37 (7) (1992) 1013–1017.
[25] F. Yates, Validation of simple and complex models in physiology and medicine, Am. J. Physiol. Regul. Integr. Comp. Physiol. 234 (5) (1978) R159–R160.
[26] C. Lloyd, M. Halstead, P. Nielsen, CellML: its future, present and past, Prog. Biophys. Mol. Biol. 85 (2004) 433–450.
[27] F.S. Grodins, Integrative cardiovascular physiology: a mathematical synthesis of cardiac and blood vessel hemodynamics, Q. Rev. Biol. 34 (2) (1959) 93–116.
[28] K. Foster, Math on the Internet, IEEE Spect. 36 (4) (1999) 36–40.
[29] P. Khodade, S. Malhotra, N. Kumar, M. Iyengar, N. Balkrishnan, N. Chandra, Cytoview: development of a cell modelling framework, J. Biosci. 32 (5) (2007) 965–977.
[30] P. Hunter, P. Robbins, D. Noble, The IUPS human physiome project, Eur. J. Physiol. 445 (2002) 1–9.
[31] A. Cuellar, C. Lloyd, P. Nielsen, D. Bullivant, D. Nickerson, P. Hunter, An overview of CellML 1.1, a biological model description language, Simulation 79 (12) (2003) 740–747.
[32] D. Chang, N. Lovell, S. Dokos, Field Markup Language: Biological field representation in XML, in: Proceedings of the 29th International IEEE Conference on Engineering in Biology and Medicine, Lyon, 2007, pp. 402–405.
[33] R. Ward, L. Pouchard, J. Nutaro, Integrative computational frameworks for multiscale digital human modeling and simulation, in: Proceedings of the International Conference on Comp. Science (ICCS 2006), Springer Verlag, 2006, pp. 814–821.
[34] P. Murray-Rust, H. Rzepa, Chemical markup, XML, and the World Wide Web. 4. CML Schema, J. Chem. Inf. Comput. Sci. 43 (3) (2003) 757–772.
[35] M. Hucka, The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models, Bioinformatics 19 (2003) 524–531.
[36] C. Bock, SysML and UML 2 support for activity modeling, Syst. Eng. 9 (2) (2006) 160–186.
[37] J. Larsson, A framework for implementation-independent simulation models, Simulation 82 (9) (2006) 563–579.
[38] M. Nuruzzaman, Modeling and Simulation in Simulink for Engineers and Scientists, AuthorHouse, 2005.
[39] S. Campbell, J.-P. Chancelier, R. Nikoukhah, Modeling and Simulation in Scilab/Scicos, Springer Verlag, 2006.
[40] A. Pop, P. Fritzson, ModelicaXML: A Modelica XML Representation with Applications, in: Proceedings of the 3rd International Modelica Conference, 2003, pp. 419–430.
[41] M. Dempsey, Automatic translation of Simulink models into Modelica using Simelica and the AdvancedBlocks library, in: Proceedings of the 3rd International Modelica Conference, Linköping, 2003, pp. 115–124.
[42] R. Hornych, M. Hurák, Sebek, MathML in Polynomial toolbox for Matlab, in: Proceedings of the International Symposium on Computer-Aided Control System Design, Glasgow, 2002, pp. 284–286.
[43] F. Tränkle, A. Gerstlauer, M. Zeitz, E. Gilles, ProMoT/Diva: a prototype of a process modeling and simulation environment, in: Proceedings of the 2nd IMACS Symposium on Math. Modelling (MATHMOD), Vienna, 1999, pp. 341–346.
[44] K.K. Bettenbrock, S. Fischer, A. Kremling, K. Jahreis, T. Sauter, E. Gilles, A quantitative approach to catabolite repression in Escherichia coli, J. Biol. Chem. 281 (5) (2006) 2578–2584.
[45] R. Waschler, O. Angeles-Palacios, M. Ginkel, A. Kienle, Object-oriented modelling of large-scale chemical engineering processes with ProMoT, Math. Comput. Model. Dynam. Syst. 12 (1) (2006) 5–18.
[46] P. Shannon, A. Markiel, O. Ozier, N. Baliga, J. Wang, D. Ramage, N. Amin, B. Schwikowski, T. Ideker, Cytoscape:, A software environment for integrated models of biomolecular interaction networks, Genome Res. 13 (2003) 2498–2504.
[47] P. Barrett, B. Bell, C. Cobelli, H. Golde, A. Schumitzky, P. Vicini, D. Foster, SAAM II: simulation, analysis, and modeling software for tracer and pharmacokinetic studies, Metabolism 47 (4) (1998) 484–492.
[48] A. Clement, Using differential properties of the green function in seakeeping computational codes, in: Proceedings of the 7th International Conference on Numer. Ship Hydrod., Vol. 6, 1999, pp. 1–15.
[49] E. Crampin, M. Halstead, P. Hunter, P. Nielsen, D. Noble, N. Smith, M. Tawhai, Computational physiology and the physiome project, Exp. Physiol. 89 (1) (2004) 1–26.
[50] M. Cabrera, G. Saidel, S. Kalhan, Lactate metabolism during exercise: analysis by an integrative systems model, Am. J. Physiol. Regul. Integr. Comp. Physiol. 277 (1999) R1522–1536.
[51] J. Kofranek, J. Rusz, S. Matousek, Guyton's diagram brought to life - from graphic chart to simulation model for teaching physiology, in: Proceedings of the 15th Annual Conference on Tech. Comp., Prague, 2007.
[52] S. Thomas, P. Baconnier, J. Fontecave, J.-P. Francoise, F. Guillaud, P. Hannaert, A. Hernandez, V. Le Rolle, P. Maziere, F. Tahi, R. White, SAPHIR: a physiome core model of body fluid homeostasis and blood pressure regulation, Phil. Trans. R. Soc. A 366 (2008) 3175–3197.
[53] B. Smith, J. Boyle, J. Dongarra, B. Garbow, Y. Ikebe, V. Klema, C. Moler, Matrix Eigensystem Routines, EISPACK Guide, Lecture Notes in Computer Science, vol. 6, Springer Verlag, 1976.
[54] J.J. Dongarra, J.R. Bunch, G.B. Moler, G.W. Stewart, LINPACK Users' Guide, SIAM, 1979.