The HOM Model: A Unified Object-Oriented Variable-Speed Human Physiology Simulation

Sanjay Manohar MB BChir MA MRCP
Department of Physiology, University of Cambridge, UK & Institute of Neurology, Queen Square, University College London, UK

Abstract

We describe an approach to the computational modelling of human systems physiology. The HOM model is a piece of software incorporating systems of equations and variables with a graphical interface. The physiology design is object-oriented and extensible, and allows calculation at variable speeds. The equations aim to be general in scope and robust. The interface is interactive, and allows on-the-fly modification of model parameters. Due to its novel structure and mode of operation, the model may be useful in teaching, research and practice of both physiology and medicine. We discuss several ways in which such models might increase our understanding, and propose an ongoing process of extension and refinement.

A: Introduction: Current approaches to modelling
- Physiology
- Variables
- Equations
- Time
- Models in teaching physiology

B: Implementation: Our choices for modelling
- Our goals
- Immutable versus immutable variables
- Environment versus body variables
- Object-oriented design
- Timing and threading
- Randomness
- Model specifications

C: Discussion: Systems modelling issues
- Anatomical versus physiological models
- Separating the plant from the controller
- Incorrect predictions
- Generality-accuracy trade-off
- Explaining physiology
- Level of modelling detail
- Conclusion

A: Introduction: Current approaches to modelling

The last 20 years have seen rapid increases in computer ownership, processing speed, memory, bus width, and advances in languages. Computational modelling of human physiology is one of the broadest and fastest developing areas in applied physiology, however by and large these computing improvements have not fully been exploited by physiology models. The goal of modelling is to characterise the body's dynamic processes in a way that will not only aid the teaching and study of normal physiology, but also the investigation of disease processes and treatments. These goals to some extent parallel the developments in the human genome project over the last 20 years, but are in some ways more challenging (Kohl, 2000). The 'physiome', as the name suggests, aims to describe the building-block elements of physiological systems and integrate their properties. As described by Kohl, establishing this information base may have an impact on drug and device development, teaching, acute decision making, long-term healthcare development, the balance between standardisation and patient-specific treatment strategies. A large amount of data has been collected for isolated systems; the more isolated and restricted the system, the more detailed and accurate is the characterisation that can be modelled.

Physiology

Several computational models of individual organ systems have been described. Our model integrates key features and variables from several models, but the equations and constants used are independently derived. Our description of the cardiovascular system is similar in character to the original Guyton model (Circ Res 35:2:159, 1974), and incorporates autonomic feedback similarly to Smith et al. (Comp Meth Prog Biomed 86:153, 2007). The respiratory components have similarities to the Olszowka and Farhi (Resp Physiol 4:270, 1968) model, with tidal breathing and respiratory control mechanisms similar to Topor et al. (Annals Biomed Eng 32:11:1530, 2004). These models make accurate predictions of the behaviour of systems in several experimental conditions. A major advantage of Topor's model is its state-dependent architecture, which we adopt throughout. In addition we incorporate a multi-compartment fluid and electrolyte model (mathematically very similar to the descriptions of Holz & Fahr, 2001), and simplified models of renal filtration and absorption, gut absorption, exercise, temperature, and sugar metabolism.

Several smaller scale physiological models underlie many anaesthetic simulators; for a review see van Meurs, Good, & Lampotang. Some of the underlying physiological principles are based on the Beneken & Rideout (1968) mathematical formulations of cardiopulmonary function and compartment models.

Variables

The standard approach is to take a physiological subsystem, and store numerical variables that correspond to the empirically measurable parameters.

A system of simple equations is then implemented, by which the future state of the model can be calculated. In addition, several non-directly-measurable parameters may be chosen, with suitable dimensions, which are used to connect the measurables. These 'hidden state' variables must be carefully selected, and we must clearly state the assumptions behind their dependencies. Unit-wise, variables can be further subdivided into instantaneous quantities (often directly measurable) and rates of change (usually evidenced by measuring the quantities over time).

Constants are necessary in any model, even if only as initial values. Constants can be categorised as either physical constants, that generally represent the choice of units, or physiological constants, which are properties specific to the organism in question.

Due to the large range of magnitudes required, variables are generally in floating-point representation. Precision can be an important factor in performance particularly because of the iterative, nonlinear and unstable nature of the system being modelled. In fact we find that 16-bit precision can produce significantly different results from 32-bit precision if larger time-steps are used.

Equations

Equations are implemented by setting the value of a variable to the result of an expression, which is obtained by performing arithmetical operations. The operations can constitute of addition, multiplication and exponentiation of the values of other variables. Due to the nature of computers, they operate at discrete time intervals. As a consequence, in the general case it is helpful for as many variables as possible to be represented as rates. This obviates the integration steps (additions on successive time intervals) that are required in time-dependent processes. There is then a natural division between the temporal equations (usually first-order partial differential equations) and simple linear equations of state. This distinction becomes apparent when implementing a non-constant time slice.

Like variables, equations can also be categorised into those that are properties of essentially physical systems, and those specific to the organism. Here physical systems include, for example*, loss of energy from the skin by sweating, where the rate of sweating is assumed to be equal to the rate of evaporation, and the latent heat is assumed to be removed from the surface of the skin. Here, physical constants would signify the physical properties of water, and the choice to represent skin temperature in degrees. The physiological system might comprise the mechanisms of sweat production. The nervous responses controlling the sweating rate can be considered as distinct, as they are generally negative feedback systems that have qualitatively different effects than the physiological processes they control (Figure 1).

Gain-based plant-and-controller design has previously been successfully used as a simple but stable model of many processes. In these models, negative feedback around an equilibrium value provides a stable base on which other perturbations can be added or subtracted. The perturbations provided can be 'external', for example, a change in air pressure; but they can also be internal, given by other controllers. A common scenario is that several controllers collaboratively determine a single variable's value. For example* stroke volume is controlled by the sympathetic system, but is influenced directly by filling pressure, which is influenced by several other controllers such as blood volume and the renin-angiotensin axis. Any system in which two controllers are coupled in this way by intermediate variables can exhibit three difficult problems: periodic oscillatory behaviour, unpredictable variability, or simple but physiologically unexpected responses.

- Periodic behaviour is normally induced when a controlled variable is far from its set-point. This is because the forces that keep it far from the set-point are of a greater magnitude, and increase susceptibility to resonance. Oscillations that are not found in vivo can be cured by adding appropriate damping terms, or low-pass filtering.
- Unpredictable variability is indeed a common feature of real organisms; however chaotic behaviour in computer models is rarely accurate or usefully predictive, and in certain experimental situations it is useful to remove these effects. Cleaner physiological responses can sometimes be produced by increasing the gain in negative feedback loops, using finer time slices, and minimising nonlinearities in the equations.
- Simple but physiologically unexpected responses usually reflect an oversight in the design of a model. These are often the most interesting and difficult to pinpoint errors, driving us to enquire exactly why the model deviates from reality. I shall illustrate this below under 'Incorrect predictions' and 'Explaining Physiology'.

In terms of variables and equations, much of the work still to be done lies in uniting disparate models, each of which deals with a subset of all possible variables. Each model may choose a different representation of a variable for its own convenience, and each has its own set of hidden unmeasurable variables.

Time

As mentioned, we generally we choose to represent quantities in a way such that the variation we are interested in lies within a manageable range. For example*, we might choose to store a local concentration gradient of calcium ions in a different format than storing total bone calcium content – leaving the differences to be dealt with in the conversion process.

Time values also vary over a huge range of magnitudes. Unlike variable values however, it is often necessary to represent time values with great precision even when dealing with long periods. That is, there are situations where we might be interested in both the sub-second variations and the monthly variations in the same variable. Most models that have been implemented so far use their own fixed relative representation of time intervals, at a scale that is suited to the process being studied.

Models in teaching physiology

Computer models of several simple scenarios have previously been devised for teaching physiology, and form a natural development from the use of equivalent circuits as analogues of physiological systems. Many authors have commented that computer simulations have improved students' understanding of physiology (Kofranek et al., 2001). It has also been noted that computer simulations reduce the numbers of animals used in undergraduate physiology teaching (Dewhurst 1995). Individual subsystems have been modelled in some detail; Fenton et al (2002) developed detailed interactive real-time Java models of cardiac electrical activity. Less detailed models have also shown to be useful; Chauvet et al. (1999) developed several smaller scale interactive programs that aid the teaching of individual subsets of equations, in an attractive way that allows replication of key experiments *in silico*. Kootsey et al. (2001) have also developed a set of general physiology Java applets that simulate individual systems in an interactive manner.

However these models may make it hard to get a feel for how systems interact in real life. Global physiology models help students to translate the principles of isolated systems into the complex interactions seen in vivo, and those involved in making medical decisions. This has motivated Kofranek et al. (2003) to build a monolithic formulation that incorporates 39 differential equations, 89 inputs and 179 output variables. This model has greater power and scope, and gives a far more realistic picture of the multifactorial nature of physiological responses. Large numbers of variables, however, can be daunting for students, and it is wise to keep them clearly organised.

By teaching initially with highly pared down simplistic models with few variables, and later increasing the number of variables, an understanding of the various levels of control can be achieved. Another advantage of graded levels of simulation is to cater for students at different levels of understanding. Advanced students, for example, might benefit from being able to see a larger set of variables.

B: Implementation: Our choices for modelling

Our Goals

We aim to construct a model that harnesses modern advances in microcomputers and programming in an interactive, real-time, graphical and modifiable environment. The physiology itself intends to be broad in scope, stable under noise, but simple enough that users can intuitively understand the variables (See Table 1).

We have chosen to begin modelling with simple equations that generalise easily. We aim to model as many diverse potential physiological scenarios as possible. With this in mind, the equations must be robust over any possible scale of value, with appropriate error-checking. With only a few variables, there is still huge potential for unforeseen scenarios, and as more variables are added, these possibilities grow at least exponentially. Therefore the equations must also aim to be robust over interpolation, i.e. small changes in a value generally yield small changes in other model variables.

The features of robustness would include the following:

1) no signs of sensitive dependence on initial conditions except where this is physiologically expected
2) making sensible predictions for previously untested combinations of values
3) with an appropriate initial state, stability of values over long periods of time
4) consistent predictions when the equations are implemented with different degrees of temporal accuracy.

***e.g. example of robustness?

Immutable versus mutable variables

Many variables' values depend on other variables, and sometimes the dependence is instantaneous. That is to say, some variables are fully determined by the value of other variables at any instant. An example* of this situation might be the atmospheric partial pressure of oxygen, which is precisely dependent on the product of percentage of oxygen and atmospheric pressure. In this situation, to alter the value of this variable directly would be ambiguous. Thus there are many variables that it does not make sense to set directly, and this is reflected in the variable design.

Environment versus body variables

Most variable values can be thought of as properties of the organism itself, or properties of its environment. This distinction is useful in that environmental variables are usually modifiable by an experimenter, whereas physiological 'body' variables are somewhat harder to modify. Also, environmental variables are not affected directly by controllers (although they might be indirectly affected by a plant). By default, they might remain constant.

Body variables can be further divided into constants and variables, but it is not obvious how the constants here might be manipulated in vivo; altering them should be permitted, but is conceptually different from environmental variables. Body constants often represent physical or chemical constraints, and some change with demographics. Examples include some set-points of controlled variables (e.g. baseline heart rate), and fixed volumes (such as respiratory dead space, or maximum stomach capacity).

Object-orientated design

In order to represent the above distinctions, we have adopted an object-oriented programming design. Object-oriented programming was developed in the 1980s to simplify the large number of ways of representing information. The idea is that each representation of information is bundled along with a set of operations acting on that information. This helps to smooth the edge between data and program, or rather, facilitates the re-use of data and code. In short, the building blocks of programs are classes, and the building blocks of data are objects; classes are the templates for objects, and objects are the concrete instances of classes.

By using an object-oriented architecture we hope that 1) our knowledge of similarities of process in many parts of the model could be captured and incorporated into the class structure, and 2) by creating a set of physiological building blocks, the class structure would itself suggest further and more elegant representations of physiological processes.

The project includes over 600 variables, but has the potential to contain thousands. The object-oriented architecture directly results in a division of variables into uniquely named and generic (unnamed) variables. This is because variables can be generated automatically using the class templates – for example*, all physiological solutions have potassium concentration. If CSF is a solution, then a variable for CSF potassium concentration will be created. Because the current set of equations only need to refer to this value indirectly (via operations on CSF as a whole), it has not been assigned a unique name. Because a plethora of solutions could be created in different contexts, it is enough to refer to these values by reference to their container object (e.g. CSF). Other variables, such as urinary potassium, have been assigned unique names as a convenience, but can still be referred to via their container.

Another direct consequence worth pointing out is that variables are sometimes created dynamically, during the simulation. This is of obvious benefit when modelling an unusual situation, or giving a drug. In this case there are many possible drugs, making it uneconomical to have pre-existing variables for each unused drug.

Organ-centric object design:
The natural way to organise an object-oriented system of physiology is system-wise, with each organ system consisting of hidden (internal) and visible (externally viewable and modifiable) variables, plus the program that manipulates the hidden variables in such a way as to produce a result in the visible variables. Should many systems need to interact with each other, they must only do so at their ‘interfaces’ – i.e. using the visible variables. In other words, each data value should ‘belong’ to a particular section of program, or vice versa. For example*, in the kidney, concentrations at various points in a nephron are internal to the kidney, in that they cannot directly influence any other organ system.
Designing an object-oriented system becomes tricky when large amounts of information are shared between large amounts of code. In the human body in particular, each organ system seems to interact with every other, and variables that might be considered intrinsic to the cardiovascular system are often frequently used by other organ systems.
A second glance shows us that interfaces between organs can, in fact, be simplified. The most common interface is the blood, which is essentially data shared between all the organs. The important information in blood includes ion and hormone concentrations, gas pressures, temperature; any organ might alter these. Additionally every organs need to know arterial and venous hydrostatic pressures. A closer look reveals that all organs have access to and from the central nervous system; however, this medium might not be best modelled this way – see below under “Anatomical versus Physiological Models”.

Finally, this method places important constraints on which variables any process can access; this can help to ensure a model maps more accurately onto real processes – see below under “Levels of Modelling Detail”. We feel that the object-oriented approach significantly eases the understanding and assembly of physiological models.

The core classes include:

- Organs, which each have a vascular resistance that determines blood flow, and a set of system-specific variables with the program to manipulate them.
- Containers, which are the elements of the compartment model. Each contains a volume plus concentrations of the simple electrolytes. The class includes a set of operations on containers, such as addition and withdrawal.
- Variables, which encapsulate a value and its units. Named variables also store the name, normal range, initial value, and whether the variable's value can be set. Each variable may belong to an organ or a container.
- Controllers, which are short pieces of code that modify a variable's value in a time-dependent manner, based on perturbations in other variables.

Timing and Threading

The goal of the timing module is to allow arbitrary time compression for modelling a single set of processes. That is, each equation must be implemented to run with an arbitrary time slice. A given amount of 'body-time' elapses between executions of any one equation. Depending on the computer speed and threading, an equation can be invoked every 10ms. The model might be running in real time mode, meaning the equation calculates what its variables values would be after 10 ms; or the model might be running in compressed time, such as '1 hour per second', meaning after 10ms the equation calculates what its variables values would be after 36000 ms.

Not all simulated processes need to operate at the same temporal resolution. For example*, the rate at which heat is exchanged between blood and skin will not need as frequent computation as, say, the increase in alveolar CO2 concentration during breath-holding. Depending on the required accuracy, heat could be calculated every 10 minutes of body-time, and CO2 concentration every second. This immediately suggests a multi-threaded approach, with each set of equations being aware of how accurate they need to be. The advantages of this include allocation of processing power to the tasks that require it, and the possibility of distribution in multi-processor architectures.

Our initial approach was that each individual physiological subsystem ran its group of equations in a single thread, with a pre-specified weighting. When a subsystem became active, it locked the physiological data it required (blocking other subsystems, but preventing concurrent modifications) – which includes common elements such as the blood.

However, as the model grew in the number of shared variables, there were several problems with this approach. Firstly, because of stringent specifications as to which variables each set of equations could access, the program became difficult to maintain, modify and extend. Secondly, the time cost

of locking and unlocking variables increased nonlinearly with the number of equations, because many equations required access to more than 2 variables. Thirdly, we encountered unanticipated oscillatory behaviour in several situations, which were in fact due to irregularities in our operating system's threading model, and its tendency to bunch together several calls within a single thread. This latter difficulty could be surmounted if a more evenly distributed threading architecture is used, or if we ensured the priorities of our threads were adhered to on smaller timescales.
These problems led to abandoning the multithreaded approach, and adopting a traditional single execution cycle. This retains a reasonable computation speed and increases predictability of results. Due to the initial design, it should be easily possible to reintroduce threading at a later stage.

We intend to capture qualitative differences in behaviour over different timescales. One issue is that, for the heart and lung, longer intervals between cycles would result in different phases of the cardiac/respiratory cycle on each calculation. This would introduce much unnecessary noise into the equations, and so when longer time-slices are used, all calculations are performed for zero-phase of these cycles. (Note that this approximation is only needed because of processor speed limitations.)

Randomness

Testing a model requires introduction of new values each time the model is run. With a highly general model, it makes some sense for this novelty to be integral to the model, rather than externally provided. To this end, minor random perturbations are introduced at a level that mimics organisms: in decision-making. When running in autonomous mode, decisions to ingest food and drink, sleep and wake, and perform exercise are made using pseudorandom numbers. The obvious disadvantage of this mode is that results are not necessarily replicable. Indeed, replicability despite noise is a good measure of how robust (robustly right or wrong) a particular prediction of the model is.
Certainly some experimental situations are best with such factors under full control. When modelling disease processes, though, it can be helpful to study models both with and without random perturbations. A further development will be direct addition of noise to specific variables' values, with controllable correlated noise models.

Model specifications

The model is written in Java and is distributed as an executable 1.5MB Jar file or a web applet. It runs on any computer with Java VM 1.4 or higher. Approximately half of the program is graphical interface, and half is physiology.

Currently the model incorporates 764 variables, of which 252 are individually named. 64-bit signed floating point numbers are used. They are organised into 11 organ systems. There are 39 modifiable feedback controllers.

C: Discussion: Systems Modelling Issues

Anatomical versus physiological models

One difficulty when creating models arises when processes (and their required variables) are assigned to organs. The physiological connections between variables span anatomical locations, as often happens for some endocrine glands and in the CNS. Many CNS mechanisms have an organ-specific regulatory function, and it is arguably better to keep these regulatory mechanisms within the main organ.

Consider for example* vomiting. The vomiting centre in the brain receives input from the gut; it performs computations on this input, and returns a volley to the gut if a threshold is reached. Since we have a system-specific reflex, surely this would be best implemented in the gut itself? If we need to model top-down factors such as psychological suppression, we may be justified in the roundabout CNS route. Similarly, if we were interested in modelling anatomical disturbances such as autonomic lesions, we would certainly need the nervous model. Another argument for the emesis centre to be modelled in the CNS module would be for the modelling of antiemetics: it is only sensible for the CNS to judge whether a drug can cross the blood-brain barrier.

In some situations, we have chosen centralised control, whereas elsewhere it is devolved to the end-organ.

Separating the plant from the controller

Physiological models are often based on controllers, where a sensor detects an abnormal value of a variable, and sets in motion a response variable to correct it. This is simple when the abnormal variable is controlled by a single variable. If a variable depends on the output of several controllers, however, there are several possible corrective measures. In this case, it is necessary to consider which of the controlling variables is the *cause* of the abnormal variable. For example*, if a tissue's perfusion is low, is this because blood oxygen is low, or blood pressure is low? Should the ventilation rate increase or the heart rate increase? Such difficulties can be solved using locally general plus globally specific strategies, for example local vasodilatation increases the tissues own perfusion, but decisions on specific correction are deferred to where more information is available (i.e. the CNS). It seems that actually, multifactorial causation can lead to a more stable situation.

Detecting an abnormal value implies that there is a normal value, a set point towards which a variable should gravitate; and that a deviation is an error to be corrected. However, with multiple controller inputs, controllers will fight each other unless the set-point itself is seen as different for each controller. Each value is then the sum of several controllers' outputs.

Incorrect predictions

It is arguable that we learn a lot more about physiology when a model makes incorrect predictions than correct ones. We must find the cause for the incorrect prediction, which requires a process of examining the variable values, their pattern of change, and how this relates to the many contributing factors in the equations.

To do this, we use a process somewhat analogous to diagnosis in medicine. In a system of several hundred variables, we have a hypothesis (differential diagnosis) as to which variable is causing unexpected behaviour, and we measure it. This in turn leads us to hypotheses about this second variable's deviation. As the development cycle continues, we learn what kinds of perturbations in which organ systems are apt to lead to a given pattern of physiological mistakes, rather like clinical intuition.

The process differs significantly from debugging computer programs, in that we do not fully understand the mechanism by which the entire model arrives at its conclusions: there are too many variables. We also do not have a full conception of what the equations ought to be, in order to produce the desired effect. In programming terms, it is more similar to disassembling and debugging a program written by somebody else. The equations supply a framework, but do not explain the whole physiological response until they are actually computed and allowed to dynamically interact.

Kofranek (2001) suggested an iterative approach to the design process of models: the implementation of a formal physiological model is verified by comparing to organisms, and the differences are used to alter the model. Unfortunately this gives us little clue as how to alter the model in order to achieve the desired result. Therefore we have several tools to assist with the process of finding the culprit equations and values when unexpected results are obtained.

- the gain constants for each controller can be modified while the program is running, allowing individual controllers to be switched on and off.
- each organ system's calculations can be turned on and off independently. When off, the organ is simply unable to alter its variables, which remain constant.

- variable values can be clamped to a given value (for mutable variables only)
- a console interface, where commands can be directly typed in while the model is running. Calculations can be made, variables can be modified, and procedures can be called
- scripts of commands can be typed, stored and executed.
- step-through timing mode, where a single time-step of a given duration can be executed, allowing examination of variables before and after a single cycle of calculation.

As an example*, I will choose a scenario that is less amenable to experimental testing. Let us consider the instantaneous removal of 500 mL of fluid from the circulation; compare removing 500 mL of saline solution versus 500 mL of whole blood, *ceteris paribus*. Now, one version of the model oddly predicts that, when saline is removed, blood volume tends to remain low, whereas if blood is removed, it rises back to near-normal levels. Why might this be? Is it correct, and if not, how can the model be amended?

The immediate differences in the saline condition are: higher blood viscosity, and higher oxygen carriage capacity. In the saline condition, it turns out that the rise in viscosity causes high capillary pressures. This has two consequences: by Starling's law, it prevents extracellular fluid from replenishing circulating volume, and in the kidney, it reduces triggering of the renin-angiotensin system. In the blood condition, the dilution by extracellular fluid causes anaemia, and a consequent vasodilatory response, further lowering blood pressure and assisting volume replenishment.
Only experiment can tell us to what extent this is true, and if it is not, evidently we must decrease the effect of haematocrit on viscosity, or possibly add vasodilation in the polycythaemic condition.

Generality-accuracy trade-off

As a simulation gets more general, the accuracy of its predictions gets less accurate. This is partly due to the intrinsic complexity of physiological systems, and means that as more variables are added to describe different phenomena, the accuracy of previously modelled variables also needs to increase. This calls for hand-coding of many exceptions to rules, and the addition of numerical 'tricks' to produce more coherent results. These 'hacks' have been carefully documented, and are usually a last resort to match experiments to predictions, particularly in cases where either the exact principles underlying the experimental results are unknown, or where these principles are too complex, fast or difficult to model.

Such situations are more common than might be expected. Here is an example of a hack*: arterial PO2 is calculated from the arterial oxygen concentration, which is expressed in mL/L of O2. Each cycle in the lungs, the dissociation curve is used to calculate the volume of oxygen absorbed. If one litre of fluid is added to the circulation, the haematocrit drops. The PO2 in the added fluid is the same as blood, but obviously the concentration is very low. As it turns out, the model does not automatically dilute the blood O2 concentration at this point, and therefore the blood becomes ‘supersaturated’ with oxygen, and it takes several cycles for the PaO2 to fall back to a normal level. This situation was corrected by proportionally lowering the arterial and venous O2 concentrations when a fluid containing no haemoglobin is added to the blood. This is programmed with loss of generality. There is no catering for an accurate pO2 in situations in which blood mixed with fluid is transfused in; that would require specification of gas pressures in every added fluid, which could be implemented at a later date. More significantly, when the blood volume is altered by other physiological processes such as ECF shifts and diuresis, we considered it unessential to recompute the oxygen concentration after very small and continuous dilutions.

Explaining physiology

It is not always easy to appreciate why physiology is the way it is. In order to understand the reasons for the organising systems in a particular way, we need a concept of what would happen if things were different. The ‘designed’ quality of carefully balanced physiological systems is only noticeable when we consider alternative designs. Modelling is one way of approaching this, with a rigour that is impossible through experiment. Using the method of separating the physical from the physiological, or the plant from the controller, we find frequently that one is carefully tailored to the other, and that the controllers’ parameters are highly sensitive to changes in many plant variables. The modelling method leads directly to a separation of the necessary from the contingent.

For example, a naïve implementation of the Starling law of capillaries leads to a description of extravasation dependent on blood pressure, plasma protein and extracellular protein. Note that that description is unaware of the total volume of extracellular fluid. As a consequence, there are many situations where this implementation generates increasing extracellular volumes, as there is no direct sensor or controller that can utilise or correct this volume.

The obvious way to correct this is the addition of compliance to the extracellular space (turgor that contributes to extracellular pressure). This could be used directly in the Starling calculations. But in fact, this pre-empts that the human body has an separate answer – lymphatic drainage. From the process of modelling, it becomes immediately clear why such a system must exist.

Level of modelling detail

It is anticipated that despite these difficulties, the level of detail of the model will increase. The more detail in the model, the stronger predictions can be made (and as described, the less likely the predictions are to be accurate). The hierarchical structure allows large numbers variables to exist without impacting on usability. For example*, the breakdown of protein into amino acids could be elaborated in great detail: a single value representing total amino acid concentration could be fragmented into the concentrations of each one, as long as the total amino acid concentration remains available to other modules. This might be useful for modelling rarer deficiency states and metabolic disorders, but can be transparent if the individual variables are within a separate class. If completeness were desired, a set of amino acid values could even be incorporated into any substance containing proteins, using the class template structure.

This raises the distinction between process-wise and functional modelling. There are some variables where it is actually better for us to calculate values differently than the body does, often because simpler models are easier to understand and can sometimes have greater explanatory power in terms of the Akaike information criterion (Akaike 1973). There is a trade-off between accuracy and economy of expression. Beneken (1998) notes in a similar vein that models can be functional or heuristic. He also lucidly describes the modelling process in terms of analogous reasoning. We would add in fact, all modelling, if viewed as analogy, is heuristic in the sense that much information is selectively discarded; the power of models lie in the economy of their heuristics.

The issue of which areas of the model need more detail is contentious, and as with any software development, priorities must be balanced with the anticipated difficulty of the task. This is notoriously unpredictable in modelling, as each change will have unforeseen knock-on effects, often in many different situations that cannot all be tested.
Tuggle (Behav Sci 23:4:271, 1978) suggested using a precise delineation of the set of phenomena to be explained, and a meaningful metric of the content of a model. In HOM, our metrics of breadth of scope and stability are clearly different from previous models, which often aim for numerical precision in a few controlled situations.
Hanna (Synthese 20:3:308, 1969) discussed the distinction between description, explanation and prediction in models in the context of experimental psychology. By considering the full range of conditions in which a model is committed to making predictions, he is led to classify these conditions into parametric variables (about which predictions are made) and 'accidental' variables. This is one clear way of setting out which kinds of variation a model intends to represent or ignore. He argues first that descriptive models maximise the probability of the observed data. Then by

considering likelihood of experimental evidence with or without a model, Hanna arrives at a definition of ‘predictive power’ of a model as the information an experimental observation provides for discriminating the prior probability of the evidence from the posterior probability. In other words, prediction is different from explanation because it endows future observations with information.

In this terminology the HOM model achieves its high predictive power by trying to minimise information transmitted by data. Unfortunately due to the continuous and temporal nature of the model, truly stochastic predictions are difficult to arrive at; we do not have good specifications as to how much noise each variable should contain (i.e. to what extent each variable is composed of parametric and accidental components). If this information is known, it would be possible to run the simulation many times to attain such predictions.

Testing and Improvement

Any model of sufficient complexity can produce vast numbers of predictions, often more than can be empirically tested – this is a consequence of the factorial combination of many interdependent variables. However only a fraction of these are useful, unexpected predictions, which deserve investigation. Thoroughly testing the model would require expertise from many fields, bringing together many facts that seem disparate, but in fact have subtle consequences for each other. We anticipate there are many experiments demonstrating phenomena which the model does not exhibit, but which it could be altered to exhibit. Miller and Walters (1974) described a method for using one common mathematical model as a point of communication between several research teams. Their model was a means of coupling together quantitative physiological data from several areas of research. With the internet, a common computerised model of this sort is conceivable.

We have therefore designed an internet feedback system whereby potential physiological ‘errors’ can be reported, along with a dump of the model’s state.

It might be useful for improving the model if we had a history of the value of each variable for every previous cycle of calculation. This would allow analysis of unreplicatable crashes, and facilitate step-by-step analysis of all problems. Unfortunately calculations show that such a history is currently unfeasible for the standard PC, requiring about 600 kB/sec storage (almost 5 Mbits/sec). However, we plan a reasonable alternative, storing just the elapsed body-time at each frame, plus the times and details of all additional events. This will permit replay and analysis of any scenario.

Expansion in the future could include further chemical metabolic details, endocrine regulatory systems including menstrual cycle, details of variation with age including paediatric physiology, and importantly more detailed models of disease.

Conclusion

The HOM simulation provides two main improvements to previous models: completeness, and extensibility. Completeness allows multiple organ systems to interact within a unified framework; extensibility offers a graphical environment in which model parameters can be modified and tested dynamically.

Thus the model could be used in multiple contexts: as a physiologist's modelling tool, a teaching aid, and a sandbox for testing hypotheses of disease and treatment.

D: References

Beneken JEW, van Oostrom JH, “Modeling in anaesthesia”, J Clin Monit 14:57-67 (1998)

Beneken JEW, Rideout VC. “The use of multiple models in cardiovascular system studies: Transport and perturbation.” IEEE Trans Biomed Eng 1968; 15: 281-289

Dewhurst D, Jenkinson L, “The Impact Of Computer-Based Alternatives On The Use Of Animals In Undergraduate Teaching - A Pilot-Study”, Atla-Alternatives To Laboratory Animals 23 (4): 521-530 Jul-Aug 1995

Fenton FH, Cherry EM, Hastings HM, Evans SJ, “Real-time computer simulations of excitable media: JAVA as a scientific language and as a wrapper for C and FORTRAN programs”, BIOSYSTEMS 64 (1-3): 73-96 Sp. Iss. SI, JAN 2002

Holz, M, Fahr, A, “Compartment modeling”, Advanced Drug Delivery Reviews 48, 249-264 (2001)

Kofranek J, Andrlik M, Kripner T, Wunsch Z, Svacina W “Multimedia simulation guides to clinical physiology”, Modelling And Control In Biomedical Systems 2003 (Including Biological Systems) : 479-485, 2003

Kofránek J, Velan T, Janicadis P and Kerekeš R.: Diagnostic and treatment of virtual patients with GOLEM – multimedia simulator of physiological function. In : Simulation in the Health and medical sciences 2001, edited by Anderson J.G., Katzper, M. Society for Computer Simulation International, San Diego, 2001, ISBN 1-56555-222-9, pp.157-164.

Kootsey JM, McAuley G, Hua L, “Presenting systems concepts in physiology and pharmacology with simulation applets in Java”, Proceedings Of The 23rd Annual International Conference Of The Ieee Engineering In Medicine And Biology Society, Vols 1-4 - Building New Bridges At The Frontiers Of Engineering And Medicine 23: 4032-4035, Part 1-4 2001

Miller NC, Walters RF, “Interactive modeling as a forcing function for research in the physiology of human performance,” SIMULATION, Vol. 22, No. 1, 1-13 (1974)

Norman J, Wilkins D, “Simulators for anesthesia”, Journal Of Clinical Monitoring 12 (1): 91-99 JAN 1996

vanMeurs, WL, Good, ML, Lampotang, S, “Functional anatomy of full-scale patient simulators”, J Clin Monitor 13 (5): 317-324 SEP 1997

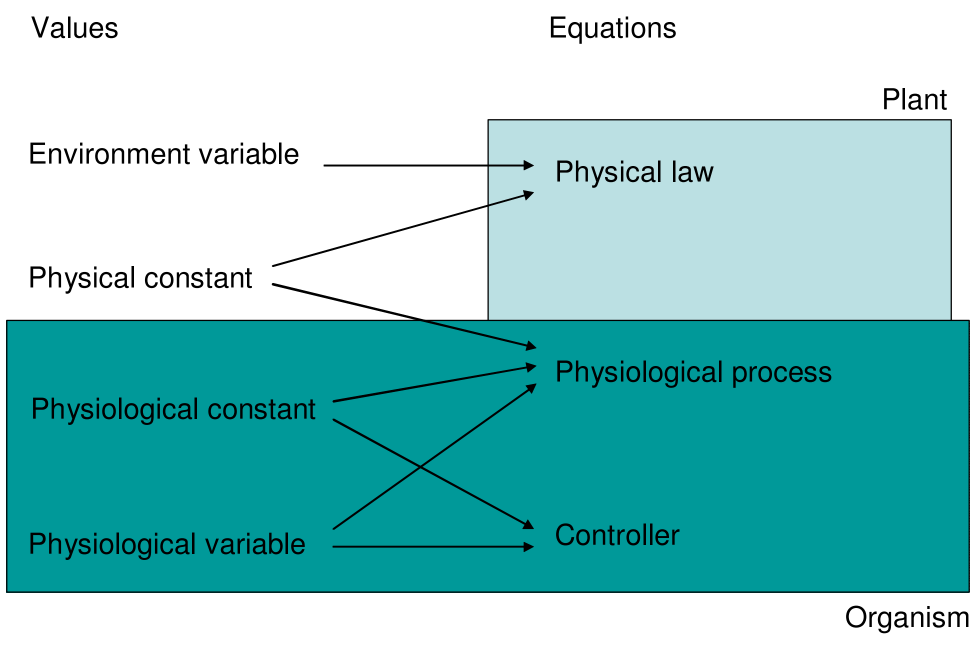

Figure 1: A way of classifying values and equations in a complex physiological model. Variables can be constant or variable for the given organism. The equations used can be regarded as physical (immutable) laws, physiological (subject to modification in disease and experiments), and controllers (feedback systems that maintain homeostasis of physiological processes). Arrows represent which types of variable are generally used as input for which type of equation.

For the Programmer:	
	Extensible
	Modular
	Physiologically debuggable
	Self-documenting
For the Student	
	Simple
	Graphical
	Dynamic/Interactive
	Robust
For the Physiologist	
	Generality for any scenario
	Completeness – system-wide
	Tweakable
	Transparent
	Intuitive representations
	Support/Feedback

Table 1: The main aims of the HOM model. As a continually evolving model, it must be designed to facilitate modification and physiological improvement.