Introduction to Modeling

Bioengineering includes aspects of both science and engineering. For more scientific questions, quantitative models can be used to test a hypothesis that requires a quantitative answer. For engineering projects, quantitative models can also be used to design or optimize something that is being built. Sometimes there is a fine line between the two, such as when we design an intervention into a biological system, such as drugs that affect insulin or thrombotic dynamics, or an orthopedic implant that must mechanically interact with the body.

Mechanistic Models. In some cases, we need to understand the mechanism of a process to answer our question, because we are testing a hypothesis about the mechanism or are trying to change system behavior by altering one or more components. In this case, it is vital that the model correctly reflect the *actual* underlying mechanism, although it is not necessary nor advised to model all the details. The mechanism must simply be correct in the details which affect the question being asked. We call such a mechanistic model a gray box model, because the mechanism is partially blacked out and partially visible. We also call this a parametric model because the parameters of the model have physical meaning that could be independently measured. For example, one might measure the size of a component, the reaction kinetics of two chemicals, a diffusion constant, or the stiffness of an elastic element.

Emperical Models. In other cases, it is not necessary to understand the mechanism of a system because we will not need to change the system. Instead, we may want to change how the system contributes to a larger system. Again, this could involve scientific questions about the role of a protein or cell in a larger physiological system. It can also involve engineering questions such as how to integrate an op-amp into a circuit, or a protein into a genetic network, to get desired behavior of that system. An alternative use of empirical models is to quantitatively compare the behavior of different related systems. An empirical model can allow a few parameters to concisely describe the difference between two data sets. The word empirical refers to knowledge that is based in observation rather than theory, so an empirical model quantitatively describes observed behavior, which is sufficient for these kinds of problems. These are also called black box models because the underlying mechanism is not visible from the model equations. The advantage of empirical models is that there is no requirement that the mechanism be known, and it is often easier to chose a model with fewer parameters.

In reality, most models have elements of both parametric and empirical models. The behavior of an enzyme may be modeled with the Michaelis-Menton Equation, which is a black box model about the enzyme structure, and could never be used to predict how point mutations would affect reaction kinetics. On the other hand, the same enyzyme may be part of a mechanistic model of a genetic network. A model of HIV dynamics may model a T-cell as a black box, but use a gray box model of the interactions between cell and virus; for example, the rate of infection of a naïve cell could be measured independently and be a parameter in the model.

Example 1: Hair cells. We know the viscoelastic response of a hair cell cilia in the inner ear in response to a movement of the tympanic membrane. We want to predict how it will respond to different types of inputs, including different frequency of vibrations. For this, a black box model of the viscoelastic response is sufficient since we are asking its response to different inputs.

Example 2: Drug delivery. We make a drug delivery particle that will slowly release a drug over time, and we characterize the release profile in vitro experimentally. Now we want to know how this release profile will affect drug levels in a patient, since we want to keep the drug in the

therapeutic window, which is above the therapeutic threshold and below the toxic. To do this, we can fit an empirical model to the release profile, and use this within a model of the pharmacokinetics of the drug in the body, which addresses the diffusion of the drug in the bodily compartments and the clearance of the drug. The empirical model of the release process is sufficient to tell us that the release profile is not good enough for this purpose. We then want to change the the particle to optimize the release kinetics. The empirical model may be able to tell us what sort of release profile we require, but would not tell us how to change the particle to obtain such a profile. For that, we would need a parametric model that describes the mechanism of controlled release correctly.

Verification and Validation. For a model to be useful, we need evidence that *the model is correct*. This is true with all methods; we need to believe that the results are not artifacts of the method used, or even caused by a mistake in methodology, but instead reflect the real system of interest. In experiments, we normally incorporate controls to show that an assay is working properly, and we often do additional experiments to determine how results in a simple system relate to a more complex one. We need to take a similar approach to computational modeling. Verification is the process of ensuring mathematical correctness, which means that the mathematical model is calculated correctly given the assumptions that were made. This requires that the assumptions were translated correctly into equations, that the equations were combined or simplified correctly to develop the model, and that the model was solved correctly, whether analytic or numeric solutions were obtained. You can think of verifications as a type of control. Like with experiments, you have higher certainty if you get the same answer in two independent studies (e.g. have two people program the same assumptions into a model), or if you can run a control in which you know what to expect because the system is greatly simplified (e.g. run the simulation with some parameters set to zero, or on a subset of the model.) Validation is the process of testing the model's ability to capture the real system, which effectively means testing whether the assumptions in the model match experimental or clinical data. The most common approach is showing that the model reproduces some known behavior. Another approach (for mechanistic models) is to determine the parameters and the model structure in independent experiments. A third method is to predict from the model some system behavior, and test this after the fact. If the model can predict something that was not used in the process of building the model, this is considered a strong validation, while reproduction of existing data (including measurement of parameters) that was used to build the model is weak validation.

Value added: Innovation/Significance/Impact. For a model to be useful, we have to *learn something new from the model*. The model might allow us to confirm or test an unknown hypothesis. Or, it might be used to help design something. If the model simply reproduces observed behavior, without a plan for how it might then be interpreted or used in future studies, it is of limited use. The usefulness often comes from predicting how the system will respond to different parameters, or identifying the limitations on parameters necessary to provide observed or required behavior. For this reason, we usually perform a systems analysis that addresses quantitative questions like these.

In summary, if a model is too close to what is already known, it has no innovation, but if it is too far from what is known, it has no certainty. The challenge in modeling is to design a model that can be validated but still provide critical new information. In this situation, modeling can be a powerful tool for experimental research or design. We will provide examples throughout this class, but the project will be your chance to plan a good use of modeling.

General Approach to Building Models

For this course, we will learn to build mathematical models and solve them numerically for the types of differential equations described below. Building a model involves translating a set of verbal assumptions into a set of the differential equations and initial conditions, as well as sometimes boundary conditions. Before we start building mathematical models, we review terminology.

Independent variables are the variables that do not depend on the model behavior.

Dependent variables are the variables that change dynamically in a system as a function of the independent variables.

The system state is the set of values of the all the dependent variables collectively and is a function of the independent variables. When we solve for the model, we find the system state.

Parameters are values that are independently determined, so they do not depend on the dynamics of the system. They are often constant, but they do not need to be, since they may change as a function of the independent variables. Unlike dependent variables, parameters cannot depend on the dependent variables. Therefore, parameters may change in a predetermined manner (fluid velocity may change with position, or an external concentration may be switched at some time), but not as a result of the system state.

Ordinary differential equations (ODEs) have one independent variable (usually time) and one or more dependent variables.

Partial differential equations (PDEs) have multiple independent variables (usually time and position), and one or more dependent variables. Thus C(x,y,z,t) may indicate the concentration of a chemical as a function of position and time.

Stochastic differential equations (SDEs) are ordinary differential equations in which the dependent variables are stochastic variables, because the equations have a stochastic term.

Which type of model to use: ODE, SDE, or PDE?

We will consider here several systems that can be modeled using PDEs, SDEs, or ODEs, depending on the assumptions that are appropriate for the conditions of the system as well as the questions we ask.

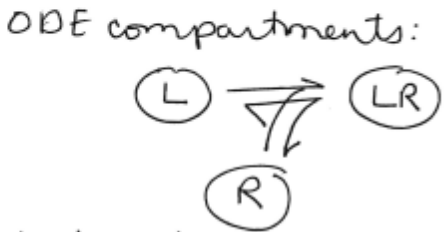
Example 1 Problem statement: Surface Plasmon Resonance (SPR) is a method that measures the change in mass of material that is very close to a gold surface. This chance in mass can be converted to the amount of bound material, so we can use the SPR to measure chemical binding reactions by unlabeled samples by immobilizing a ligand on the gold surface, and flowing through a receptor in solution, because the receptor-ligand complex has more mass than the ligand L. Thus, response is proportional to the amount of complex formed. By fitting models to this response curve, we can calculate rate and equilibrium constants for the binding reaction. We call the concentration of uncomplexed ligand and receptor L and R respectively, and the concentration of complex C. L_0 (in molar, M) is the total concentration of ligand on the chip, and R_0 (in M) is the concentration of receptor that is pumped through the device. If the binding reaction can be modeled as simple one-state binding, we have two reaction equations:

Binding reaction: $R + L \rightarrow C$, with rate constant k_{on}, in units $M^{-1}s^{-1}$.

Unbinding reaction: $C \rightarrow R + L$, with rate constant k_{off}, in units s^{-1}.

We want to model how much complex, C, forms as a function of time?

ODE models. If we have large enough numbers of molecules to neglect intrinsic stochastic noise, and transport is sufficiently fast relative to reaction rates to neglect spatial variations in concentration, then we can use an ODE model. These are much faster to build and to solve than PDE or SDE models, so should be used if these assumptions can be justified. If don't know if the assumptions are justified, then solve the ODE model, and do a sanity check with the solution against these assumptions, or compare it to data to see if it fits without the complications of the PDE or SDE model.

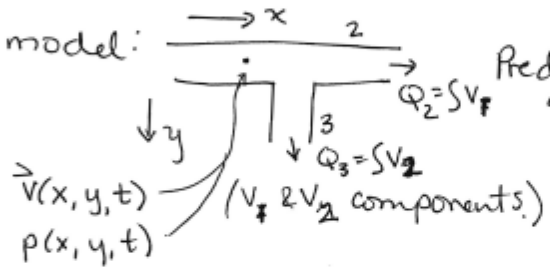


PDE models. If the receptor binds to immobilized ligand faster than it is replenished by convective flow in the device and by diffusion, then the concentration of receptor near the surface will be depleted, which will affect the reaction rate. In this case, we need to model the transport (convection and diffusion) as well as the reaction. This requires a model that describes the concentration of receptor and ligand as a function of position as well as of time, which means a partial differential equation (PDE).

SDE models. If you are using a new technology that detects single particle interactions, you may need to model each molecule individually to compare the results to data. If you need to model the movement of single molecules through space, then you will use Brownian Dynamics simulations, which model the stochastic position (x(t), y(t), z(t)) and state of an individual particle. Brownian dynamics provide a discrete stochastic version of transport/reaction PDE equations, and will converge on the PDE solution with large enough numbers, at least for linear systems. If your new technology does not necessarily detect single molecules, but has nanoliter volumes that involve only 100 or so molecules, then the intrinsic noise due to the stochastic rates of reaction may have a greater effect on your experimental data than will the noise introduced by your measurement methods. If you can assume mixing within this small compartment, then you can use use stochastic reaction equations to model a stochastic version of the compartmental ODE model. Both types of stochastic simulations are forms of stochastic differential equations (SDEs).

Example 2 Problem statement: We may want to design an object that will bend like a cantilever with a desired spring constant. If the cantilever has a regular geometry, we can look up or derive the analytic equation that relates the spring constant to the length and flexural rigidity of the beam, so the design is simple. However, if the cantilever must have a complicated geometry for other reasons, there is no analytic solution and we need to use PDEs to calculate the equilibrium state of the cantilever in response to external forces, by simultaneous solution of the stress-strain equations for each small element in the cantilever. The solid mechanics equations in COMSOL can be used for such calculations.

Example 3 Problem statement: A third example is fluid flow. If we have a branched microfluidic channel, we may model this with an ODE model after estimating or measuring the resistance and capacitance of each branch of the channel. If two branches have the same resistance (R2 in the figure to the right), then the flow through each channel will be the same, regardless of the geometry of the system. However, if one branch of the channel goes straight and the other branches to the side, inertia will favor the straight channel. The ODE model assumes that this effect is minimal compared to the resistance, but a PDE model of fluid flow would test this assumption and calculate the actual flow. Similarly, our ability to estimate resistance in a channel is limited to simple geometries. Thus, to calculate the actual flow profiles in microfluidic channels, we need to simulate the Navier-Stokes equations, which models the pressure (scalar) and velocity (tensor) at all points.



Simple Verification Tools

As you build your model into mathematical equations, you should verify the equations, which means check them for mathematical correctness, and check that they match the assumptions. Actually, the main tool for verification is to perform a series of tests for mathematical incorrectness. If you find any errors, you can correct them before going further, and otherwise, you can proceed with much more confidence.

A conceptually simple but time consuming verification test is to have two people perform the same calculation twice, to make sure they get the same thing. If they don't, you know at least one of the calculations is incorrect. In class, you can compare your work to that of your peers in lab. However, in your final project and in real research, this costs double the time, or even triple if the results are different and a third version is needed. So here are some short cuts. Please learn how to use these in your weekly labs so you will be proficient by your final project.

1. Model Completeness: At the model building step, make sure that you have …
    a. Used every assumption you listed at least once in deriving the equations.
    b. One differential equation for each dependent variable
    c. One initial condition for each dependent variable.
    d. One boundary condition equation for each dependent variable on each boundary (For PDEs).
    e. All parameters are defined.
2. Dimensional Analysis: Dimensional analysis is simply determining the dimensions of each term in each equation, and making sure that they are the same within an equation. Recall that a term is anything that is added or subtracted, on either side of the equal sign. Dimensional analysis is performed at both model building step and the solution step.
3. Expected solution values. There are two times you probably know the values, and you should check that your solution matches these.
    a. At t = 0, you know the system should be the initial conditions (unless this is complicated by step function inputs)
    b. At t = infinity, you know the system will reach the steady state solution (if one exists), which can be calculated by setting all derivatives to zero and simultaneously solving the resulting algebraic equations for the variables.

Numeric vs Analytic Solutions

Once we have built a model by writing the (ordinary, partial or stochastic) differential equations and identified initial and boundary conditions as appropriate, we can solve the model analytically or numerically.

Analytic Solutions. An analytic solution has the form of an equation for each dependent variable as a function of the independent variables. Analytic solutions are powerful because they are true for all values of the parameters, and thus analytic solutions are the focus of classes you may have taken. However, many differential equations cannot be solved analytically. In some cases, it is possible to make assumptions to obtain approximate analytic solutions that apply in certain conditions. For example, steady state solutions are approximately true as long as time is greater than some characteristic value. Linearization of nonlinear ODE equations are approximately true for values of the variables near some values. Some relationships between parameters can lead to simplified forms of equations in which some physical process have no significant effect on the system behavior. Even if the assumptions necessary for an analytic solution are not realistic, these solutions can be used to verify a numeric solution.

Numeric solutions. However, many or even most biological problems involve complex models that either require too much time and experience for practical solutions, or simply cannot be solved analytically. These can be solved numerically using a computer and various software packages. However, a numeric solution requires the user to identify specific values for all parameters, and the solution is only true for these values. For this reason, numeric solutions provide much less information and are less powerful than analytic solutions. Moreover, numeric solutions are only approximate, so it is necessary to ensure that numeric artifacts are not affecting the conclusions you draw from them. Even when analytic solutions are possible, a numeric solution is useful to verify the analytic solution.

Graphical Solutions. Both numeric and analytic solutions can be plotted graphically to show the values of the dependent variables as a function of the independent variables. Like numeric solutions, a graphical solution only applies to a specific set of parameters.

Methods for numeric solutions.

Ordinary, Partial and stochastic differential equations can be solved by some method of numeric integration. In numeric integration, we know the initial conditions and rules for how variables change over time, and we use this to estimate the position at some later time step. There are many different algorithms for integrating different kinds of equations, and different ones are more efficient in different conditions. The theory, details, advantages and disadvantages of the various numeric methods for integration is beyond the scope of this class. Instead, we provide tools to find the right balance of accuracy and efficiency. That is, we teach the verification tools to ensure that your solution is sufficiently accurate to support your conclusions. We also provide the basic information to guide you to find more efficient algorithms for your problem if the required accuracy is too expensive.

Verification of numeric accuracy. It is common that a numeric solution will look obviously segmented or otherwise uneven when graphed, which strongly suggests numeric errors. This will usually not affect your conclusions, but should be addressed to avoid possible misinterpretation of the figures. A less common issue is that the approximate solution can completely diverge from the exact solution, which will affect your conclusions. If you see segmentation, oscillation,

or any counterintuitive result, you will probably try to fix it. However, some numeric artifacts may not be obvious, so **you should verify your numeric solution regardless of the appearance of your solution**. To verify that nothing in the appearance of the figures or other results reflects an artifact of your integration method, you should solve the problem several times with different algorithms and different tolerance. When you find a set of solution methods that give the same result, you can trust it, and use the most efficient of these. This verification only addresses the choice of numeric method for this problem, not whether you have developed and implemented the model correctly. The latter is more challenging and will be addressed later.

Accuracy of integration vs output. Make sure you distinguish between the accuracy of the underlying integration and the number of output values that are saved when the integration is complete. For example, MATLAB's ODE solvers may give so few output values that the graph looks segmented, but each value is already very accurate. Increasing the tolerance won't smooth this curve. Instead you must ask for more output values, which can be done by providing a specific time vector as input.

First-order integration. Consider a differential equation in one variable. This will have the general form $\frac{dy}{dt} = f\big(t, y(t)\big)$, where f is a known function. While we do not have a general form for $y(t)$, we do know one value $y(t_0)$ from our initial conditions. We need an algorithm to estimate $y(t_0 + \Delta t)$ from $y(t_0)$, where Δt is a small increment in time. We can then iterate this process using that new value of y instead of the initial condition.

Recall that the Taylor expansion of a function to obtain higher-order approximations for the integration step. Recall that the Taylor expansion for a function y near t_0 is:

$$y(t_0 + \Delta t) = y(t_0) + \Delta t \cdot \frac{dy}{dt}(t_0) + \frac{1}{2}\Delta t^2 \cdot \frac{d^2y}{dt^2}(t_0) + \cdots$$

Then recall that we have a known equation for the derivative, $\frac{dy}{dt} = f\big(t, y(t)\big)$, so we immediately have all the information needed for a first-order approximation of $y(t_0 + \Delta t)$:

$$y(t_0 + \Delta t) \sim y(t_0) + f\big(t_0, y(t_0)\big) \cdot \Delta t$$

If Taylor expansions make you nervous, you can derive the same equation by rearranging the definition of a derivative that you learned in differential calculus: $\frac{dy}{dt} = lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t}$, or $\frac{dy}{dt} \sim \frac{y(t+\Delta t) - y(t)}{\Delta t}$. Either way, we can estimate the new value of y from a value of y at a previous time point and the expression for the derivative function. The estimate becomes more accurate when Δt becomes smaller. Most of the problems we solve involve multiple dependent variables. The process is identical, but y is now a vector of dependent variables and f is the vector of derivative equations.

Tolerance and dynamic time stepping. In any robust algorithm, the user does not define a time step. Instead, the user defines the error tolerance, and the algorithm adjusts the time step to meet this tolerance. At each integration step, the algorithm calculates a value for f, the vector of derivatives. If any element of f changes more than the required error tolerance before and after the last step, then the last step is rejected and repeated with a smaller Δt. The tolerance may be defined in absolute terms to compare with the change in each function, or in relative terms to compare with the fractional change in each function. If instead all elements are far below the error tolerance, the step may be accepted, but a large Δt will be tried next time. If the error is just

within tolerance, then the same Δt can be used for the next step. The relative tolerance is generally more robust, since it is insensitive to the units chosen for the variables. However, there are situations where a relative tolerance will slow the simulation unnecessarily in certain conditions.

Higher order integrations. If we can determine $\frac{d^2y}{dt^2}(t_0)$, we can use the second order approximation, which is more accurate for larger Δt. Many algorithms use higher order approximations even though only the first order derivative vector, $\frac{dy}{dt}$, is supplied by the user. Each step of the algorithm must estimate the higher order derivatives by comparing the estimates at several small values of Δt, so each step is much slower in these algorithms. However, the algorithm then uses a large value of Δt for the next estimate, which can make it much faster. The actual improvement in efficiency depends on how much better the higher order approximation is. For example, for a true 2nd order polynomial, the 2nd order integration will give an exact solution for infinite Δt, so would be a huge time saver.

Stiff systems and algorithms. For some systems of differential equations, the solution using standard algorithms is inaccurate unless extremely small step sizes (Δt) are used. For nonstiff problems, the required step size is small only when the solution is changing rapidly, while for stiff problems, the solution is somehow numerically unstable so that small step sizes are needed even when the system changes slowly. Alternative algorithms have been developed that tend to be more robust for stiff problems. To find these algorithms, search the documentation for the software you use. If your problem is solving very slowly, you should abort the calculation if needed, and try again with a stiff solver.

Summary. Know the terms underlined and these concepts.

1. In mechanistic, Parametric, or gray box models, the equations and parameters have physical meaning.
2. In empirical or black box models, the equations and parameters have no physical meaning but should succeed in predicting behavior.
3. Verification is the process of ensuring that the model and its solution are mathematically correct implementations.
4. Validation is the process of ensuring that the assumptions of the model are appropriate for the questions being asked, by measuring parameters, fitting the model to data, or testing predictions of the model. Validation increases the level of certainty.
5. Value added is anything we learn from the model that is not known already from the validation experiments.
6. The system state is the vector of dependent variables, which are identified by solving the differential equations for all relevant values of independent variables. The Parameters can be constant or algebraic equations of the independent variables.
7. All types of differential equations can have multiple dependent variables. Ordinary differential equations have only one independent variables, while partial differential equations have more than one. Stochastic differential equations have stochastic terms in the differential equations.
8. Analytic solutions are more powerful than numeric because the former are true for many parameter conditions, and the latter for only one set of parameters.
9. Numeric solutions are possible for a much wider range of problems than analytic solutions.
10. Accuracy is the closeness of a numeric approximation to a true solution. The accuracy depends the size of the time step, which in term is determined by the tolerance, which is the amount of relative or absolute error allowed at each time step. Accuracy also depends on the form of the differential equations being solved and the algorithm used.
11. Efficiency is how fast a simulation runs, and is the inverse of cost. Efficiency and cost depend on the equations, tolerance, numeric algorithm, and even hardware.
12. If your curve looks rough, you may need to save more output values.
13. Numeric algorithms for ODEs are often classified by order and whether they are stiff. Different algorithms have the best trade-off between accuracy and efficiency for different problems.
14. **Simplest verification tools are:**
    a. For the differential equations, make sure
        i. the model is complete (one equation, IC, and right BC for each dependent variable)
        ii. perform a dimensional analysis (make sure all terms have the same units).
    b. For a numeric solution, ensure that the solution is robust with respect to changes in the numeric algorithm or tolerance.
    c. For a numeric or analytic solution:
        i. Do a dimensional analysis
        ii. make sure t(0) matches the initial conditions,
        iii. make sure t(infinity) matches the steady state solution, if one exists (which is found by setting all derivatives to zero.)