Chapter 23

Mathematical Modeling of Photo- and Thermomorphogenesis in Plants

Gabriel Rodriguez-Maroto, Pablo Catalán, Cristina Nieto, Salomé Prat, and Saúl Ares

Abstract

Increased day lengths and warm conditions inversely affect plant growth by directly modulating nuclear phyB, ELF3, and COP1 levels. Quantitative measures of the hypocotyl length have been key to gaining a deeper understanding of this complex regulatory network, while similar quantitative data are the foundation for many studies in plant biology. Here, we explore the application of mathematical modeling, specifically ordinary differential equations (ODEs), to understand plant responses to these environmental cues. We provide a comprehensive guide to constructing, simulating, and fitting these models to data, using the law of mass action to study the evolution of molecular species. The fundamental principles of these models are introduced, highlighting their utility in deciphering complex plant physiological interactions and testing hypotheses. This brief introduction will not allow experimentalists without a mathematical background to run their own simulations overnight, but it will help them grasp modeling principles and communicate with more theory-inclined colleagues.

Key words Mathematical modeling, Ordinary differential equations (ODEs), Law of mass action, Systems biology, Dynamical systems, Numerical simulation, Bifurcation analysis, Model building, Simulated annealing

1 Introduction

After the discovery that warm temperatures modify light-grown seedling architecture and the coining of the "thermomorphogenesis" biological term, much recent work has focused on understanding the effects of both light and temperature on plant development [1]. To disentangle the complex network of molecular actors involved in sensing and responding to light and temperature, development of mathematical models is a very helpful asset. In this chapter, we outline the basic procedure to develop a

Authors Gabriel Rodriguez-Maroto and Pablo Catalán are the co-first authors for this chapter.

Meng Chen (ed.), *Thermomorphogenesis: Methods and Protocols*, Methods in Molecular Biology, vol. 2795,
https://doi.org/10.1007/978-1-0716-3814-9_23,

mathematical model of plant systems biology. We will focus on systems of ordinary differential equations (ODEs), and in what follows, we will explain the basic concepts for their derivation.

Mathematical models help us understand our system of interest by allowing us to outline its temporal behavior given certain initial conditions. They also help to test different hypotheses and check if they are consistent with our data or our previous knowledge. Finally, once they are correctly built, they allow us to make predictions about the behavior of our system outside the range of experimental conditions previously measured.

Modeling is an iterative process. First, several hypotheses about the system's behavior are made. Then, a combination of those hypotheses, the available experimental data, and theoretical knowledge (physical and chemical laws) are combined to build the model. The next step is to verify whether the model is able to capture and mimic the experimental results. If the model's results match the observed data, the model has been successfully built. On the other hand, if there are inconsistencies between the model's results and the experimental ones, we will have to review the initial hypotheses and adjust the model. In fact, this process can be done even when the results match. For instance, it might be possible to reduce the complexity of the model without omitting any biologically relevant parts by simply ignoring certain variables, grouping parameters, or simplifying some intermediate steps.

The primary objective is not to develop a mathematical model that incorporates the maximum number of variables. Even if such a comprehensive model were feasible, deriving biological insights from it would prove exceedingly challenging due to its inherent complexity [2]. Instead, what mathematical modeling pursues in biology is to develop a model able to capture the essence of the significant interactions, thus providing an outcome that can be fully understood from a biological perspective.

A mathematical model can be defined as a set of equations describing the behavior of a system. When modeling plants' responses to light and temperature, the state variables will very likely represent the abundance of specific molecular species as a function of time, as, for instance, would be the concentration of active phytochrome B (phyB) in the nucleus [3]. If the biological system under study is a biochemical network, the components of that network will be the various molecules involved, and the interactions will correspond to the chemical reactions occurring between them. Assuming we know which chemical reactions occur, they can be mathematically expressed as [4]:

$$R_j : \sum_{i=1}^{n} \alpha_{ij} X_i \xrightarrow{k_j} \sum_{i=1}^{n} \beta_{ij} X_i,$$

where R_j represents each of the reactions included in the network; α_{ij} and β_{ij} are the stoichiometric coefficients determining the number of molecules of reactants and products (respectively) involved in the reaction; the k_j are the reaction constants, and X_i identifies each of the species participating in the reaction. The left- and right-hand side terms represent the reactants and products, respectively, and the arrow indicates the direction of the reaction.

Once we have identified the chemical reactions, the next step is to derive the mathematical equations describing the temporal behavior of the system. From a mathematical point of view, this is equivalent to finding the temporal derivative of whatever magnitude is used to describe the state of the system's molecules. As we will later see, these rates depend on the species participating in the reaction, but they can also be functions of environmental conditions, such as light or temperature: many reactions happen only during the daytime [5], or reaction rates can be accelerated when temperature increases [6].

The magnitude we will use to describe the state of the system is the concentration of molecules of interest [2]. This way, the mathematical model will consist of a set of ODEs describing the temporal evolution of the concentration of molecules of each species involved in the network (*see* **Note 1** for an alternative modeling framework). Furthermore, when modeling chemical reaction networks, two assumptions are frequently considered. The first is the lack of spatial dependence in the reaction rates, which can be justified on the grounds of a homogeneous distribution of the reactants. The second assumption refers to the use of molecular concentrations instead of discrete numbers of molecules. This is possible as long as the number of molecules is high enough, and it is designated as the *continuum hypothesis* [2].

If we can properly justify the two previous assumptions, the equations modeling the overall network will arise from the application of the Law of Mass Action. This law states that the rate of any chemical reaction is proportional to the product of the concentrations of the reactants [2, 4]. Consider, for instance, the simple reaction:

$$A + B \xrightarrow{k} C.$$

The rate of production of the species C can be easily obtained by applying the Law of Mass Action, and it is equal to $k[A][B]$, where $[A]$ and $[B]$ denote the concentrations of each reactant, and k is a constant of proportionality (rate constant). Importantly, the units of this rate constant will depend on the number of reactants in each reaction, while the number of products does not play any role in the reaction rate. A further important factor to take into consideration is the kinetic order of each reactant [2], which sets the exponent to which each concentration has to be powered. In the previous reaction, the kinetic order of A and B is one. If we had the reaction:

$$2A + B \xrightarrow{k} C,$$

the kinetic order of A would be two, and hence the reaction rate of C would be expressed as $k[A]^2[B]$.

Our aim is to obtain an ODE describing the temporal variation of the concentration of the different species participating in the reaction. Thus, time will be the independent variable, and the species' concentrations will become the dependent variables. In the case of a species participating in a single reaction, what we chemically called the "reaction rate" is mathematically nothing but a temporal derivative. Thus, the reaction rate of the production of C is just the temporal variation of C's concentration:

$$\frac{d[C]}{dt} = k[A][B].$$

Note also that if a species participates in several reactions, the temporal derivative will be obtained from the addition of the different reaction rates.

In studying plants' responses to light and temperature, we will need to model the transcription and translation of genes of interest, as well as the protein-protein and protein-gene interaction events comprising the gene regulatory network (GRNs). The general scheme follows the same guidelines explained above, but for the sake of clarity, we will provide some examples of modeling GRNs.

Basically, what makes these GRNs different from the previously explained chemical networks is their complexity. Instead of a set of single reactions, we have complex chains of reaction processes. For instance, both transcription and translation involve a very large number of individual biochemical reactions. However, it is not just a matter of the high number of reactions involved—these also occur in metabolic or pure chemical networks—but the fact that many of those reactions are not fully characterized [7]. This clearly prevents us from knowing the exact chemical reaction that takes place, a main feature that differentiates biological systems with respect to other chemical networks.

In order to circumvent this problem, we often make the simplifying assumption that both transcription and translation happen at phenomenological—"effective," in modeling jargon—rates, therefore modeling more abstract reactions. As such, the gene expression description will not be as precise as that of the previous biochemical networks—it will be more coarse-grained—but it will still be useful and manageable. However, detailed models of these processes also exist, see for example [8].

A second problem associated with modeling gene regulatory networks has to do with the *continuum hypothesis*, since often the number of molecules involved in the regulation of gene expression is not that high. For instance, in the case of proteins regulating gene expression, the number of copies per species is in the range of several hundred or even less [9]. Regarding encoding genes, the numbers are even lower—rarely more than dozens and typically just one or two. Surprisingly, the mass-action formalism is still applicable in systems with low numbers of molecules; however, this entails a small change in the perspective of the model [2]. Now, the differential equations should be understood as descriptors of the average behavior of the system over a large population of cells. By contrast, to capture the behavior of individual cells, we would better employ a different modeling scheme (*see* **Note 1**).

2 Materials

In order to perform numerical simulations of the model, you will need a computer. Personal computers (including laptops) are enough for most tasks, while access to high-performance computing clusters may be necessary for specific aspects such as parameter fitting, especially if the model is very complex or involves many parameters.

Numerical simulations of ODEs can be written in most programming languages. We favor C++ and Python 3, both open source and available for free, and MATLAB (Mathworks, Inc.), which requires a license. Tutorials on how to use these languages and how to simulate dynamical systems are readily available online [10–12].

We utilize AUTO [13] for bifurcation analysis, while Mathematica (Wolfram Research) aids us in handling complex analytical calculations.

3 Methods

In this section, we report on how to build a mathematical model from scratch.

3.1 Read the Literature

To start building the model, you will need to investigate the known interactions between the molecular actors involved in your system of interest, by performing a thorough literature review. In modeling plants' responses to light and temperature, you will probably have to study the effects of photoreceptors (typically phytochromes, but also cryptochromes or others [14]) on your system. You will also probably have to include the effect of the circadian clock [15]. Identify the main genes and proteins involved in your process. Write down all relevant interactions, as well as those that you hypothesize are possible but have not been measured. And, of course, do not reinvent the wheel: there are several beautiful models out there [16–19]. Take inspiration from or recycle parts of published models as needed. Building a mathematical model from the gathered information is the key part of the procedure, and the most difficult one. You will have to turn the previously noted interactions into ODEs. This can be done either using a bottom-up or a top-down approach.

3.2 Build the Mathematical Model Using a Bottom-Up Approach

Start by writing down the chemical reactions in which all the molecular actors are involved: transcription, translation, dimerization, degradation, and so on. Use the Law of Mass Action to derive a system of ODEs for each molecular species (these will include intermediate complexes as well as the molecules of interest). By inspecting the scales of the different parameter values, you can introduce steady-state approximations that simplify the system. For example, the process of a transcription factor binding to DNA is generally assumed to occur faster than protein translation. Consequently, the equation representing a TF-DNA complex can be considered to be in equilibrium. A simple example is as follows: we know that *constitutive photomorphogenic* 1 (COP1) is involved in degradation of both *phytochrome* B (phyB) and *early flowering* 3 (ELF3) [20, 21]. When we started to model this interaction, we hypothesized that phyB and ELF3 might be competing for a finite pool of COP1, so that higher levels of ELF3 would result in less phyB degradation since ELF3 would be sequestering COP1. In terms of reactions, this translates into:

$$E + C \xrightarrow{k_1} D_1 \xrightarrow{k_2} C,$$

$$D_1 \xrightarrow{k_{-1}} E + C,$$

$$B + C \xrightarrow{q_1} D_2 \xrightarrow{q_2} C,$$

$$D_2 \xrightarrow{q_{-1}} B + C,$$

where E represents ELF3, C is COP1, B is phyB, D_1 and D_2 are the respective ELF3-COP1 and phyB-COP1 complexes, and k_1, k_{-1}, k_2, q_1, q_{-1}, q_2 are the corresponding rate constants of the reactions. For simplicity, we can assume that both ELF3 and phyB will be created at rates p_E and p_B, respectively.

Translating this into equations, we obtain:

$$\frac{d[E]}{dt} = p_E - k_1[E][C] + k_{-1}[D_1],$$

$$\frac{d[B]}{dt} = p_B - q_1[B][C] + q_{-1}[D_2],$$

$$\frac{d[C]}{dt} = -k_1[E][C] + (k_{-1} + k_2)[D_1] - q_1[B][C] + (q_{-1} + q_2)[D_2],$$

$$\frac{d[D_1]}{dt} = k_1[E][C] - (k_{-1} + k_2)[D_1],$$

$$\frac{d[D_2]}{dt} = q_1[B][C] - (q_{-1} + q_2)[D_2],$$

where the brackets, as before, indicate concentrations. Considering that the formation and break-down of these protein complexes is fast, we can assume that the derivatives of $[D_1]$ and $[D_2]$ are close to zero (the steady-state assumption), and therefore express their concentrations in terms of E, B and C. Also, the sum of the last three equations is zero. This implies that the sum of the concentrations of C, D_1 and D_2 is constant and equal to the initial concentration of COP1, $[C]_0$ (conservation of mass). After some algebra, we can express the change in $[E]$ and $[B]$ by only using their rate constants and $[C]_0$:

$$\frac{d[E]}{dt} = p_E - \frac{\frac{k_1}{k_{-1}+k_2} k_2 [C]_0 [E]}{\frac{k_1}{k_{-1}+k_2}[E] + \frac{q_1}{q_{-1}+q_2}[B] + 1},$$

$$\frac{d[B]}{dt} = p_P - \frac{\frac{q_1}{q_{-1}+q_2} q_2 [C]_0 [B]}{\frac{k_1}{k_{-1}+k_2}[E] + \frac{q_1}{q_{-1}+q_2}[B] + 1}.$$

Figure 1 shows the comparison between the true solution (solid lines) and the approximate solution (dashed lines). Note that, after an initial transient, both lines quickly converge once the intermediate complexes reach equilibrium.

3.3 Build the Mathematical Model Using a Top-Down Approach

When multiple molecular actors are involved and, more importantly, the details of their molecular interactions are not well known, an alternative approach to model building is to write interactions from scratch. For instance, if we know that the expression of the *phytochrome interacting factor* 4 (PIF4) target *arabidopsis thaliana homeobox2* (ATHB2) is inhibited by phyB [22], ELF3 [23] and *elongated hypocotyl* 5 (HY5) [24], we can write the ODE for ATHB2 as follows:

$$\frac{d[\text{ATHB2}]}{dt} = \frac{p_P[P]}{1 + p_P[P] + p_E[E] + p_B[B] + p_H[H]},$$

where $[P]$ is the concentration of PIF4 and $[H]$ is the concentration of HY5, the p_X indicate the binding strength of species X, while the rest are as above. Once you get a sense of how these models work, you can come up with a model based on these types of equations without all the intermediate derivation.

In many cases, the top-down approach results in simpler equations than the ones obtained using the bottom-up approach. For instance, when we derived our hypocotyl growth model [25] in the

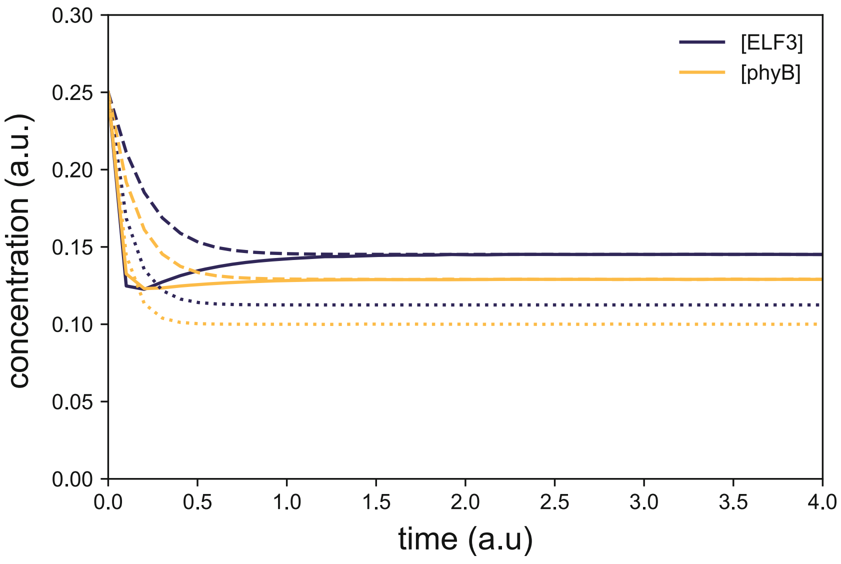

Fig. 1 Time evolution of ELF3 and phyB concentrations, using different equations. Solid lines represent the complete numerical solution, while dashed lines represent the approximate solution using the steady-state approximation for the intermediate complexes. Dotted lines show the solution when there is no competition for COP1 between ELF3 and phyB. Parameters: $p_E = 1$, $p_P = 1.2$, $k_1 = 10$, $k_{-1} = 5$, $k_2 = 4$, $q_1 = 9$, $q_{-1} = 3$, $q_2 = 6$, $[E](0) = 0.25$, $[B](0) = 0.25$, $[C](0) = 2$, $[D_1](0) = 0$, $[D_2](0) = 0$

example of the previous section, we soon realized that the ODEs had too many parameters. A model without considering the competition between ELF3 and phyB for COP1 interaction was actually equally capable of reproducing the experimental results. The equations for ELF3 and COP1 nuclear abundance without such a competition effect become:

$$\frac{d[E]}{dt} = p_E - \frac{k_1}{k_{-1} + k_2} k_2 [C]_0 [E],$$

$$\frac{d[B]}{dt} = p_P - \frac{q_1}{q_{-1} + q_2} q_2 [C]_0 [B].$$

These are plotted in Fig. 1 as dotted lines. Note that while this solution is not the same as the complete (bottom-up) model, for high enough values of $[C]_0$ it is close to the experimental data. More importantly, the qualitative behavior of the curves is very similar, which allows fitting the experimental data with these simplified equations (*see* Subheading 3.7 and **Note 2**).

Once you have decided on a final set of ODEs, you should study the properties of your dynamical system. In order to do this, you will have to use a combination of analytical and numerical methods. A good starting point is to just run some simulations of the equations, playing with the parameter values, to get a feel for what the model can do. By studying the properties of the model, you will be able to see if the hypotheses you used for developing it are consistent or not. If not, you should now modify the equations in an iterative process until you get the desired results.

3.4 Find the Fixed Points of Your System

In many cases, it is very useful to find the fixed points of the system of ODEs, i.e., the set of points that make all derivatives equal to zero. If they exist and are stable (more on stability in the next subsection), these are the stationary values where the system variables will settle once the initial transients have receded. If the system is simple enough, you will be able to obtain analytical expressions for these. More frequently, however, the nonlinearities of the system will prevent you from obtaining closed analytical expressions. In this case, you can try testing approximations for when some parameters are small or large using methods from perturbation theory [26], or find the fixed points by solving the nonlinear algebraic equations numerically. Often, the initial exploratory study of the system through simulations (more on how to do this below) will find sets of values where the variables stabilize.

Studying fixed points means focusing on the system's long-term behavior, excluding its transient states. In many cases, transients are not relevant for our purposes, and so the analysis of the model will be restricted to the final equilibrium. In other cases, we may also be interested in the transient behavior, and numerical simulations will then be necessary. Finding analytical solutions, that is, closed formulas, for the time dependence of the variables is almost for certain an intractable problem for most models in plant systems biology.

3.5 Study the Stability of the Fixed Points

The fixed points we do obtain might be stable, meaning that the state of the system robustly stays at the fixed point once it has been reached. It is important to study this stability, not only to find out whether the fixed point is a valid stationary state of the system but also to determine if more complex behaviors, like multistability or oscillations, may arise. This is usually done by calculating the eigenvalues of the Jacobian matrix associated with the system [27]. When all the eigenvalues of the Jacobian evaluated at a fixed point have negative real part, the fixed point is stable; otherwise, it is unstable. As before, you will sometimes be able to do this by hand or with the help of a symbolic programming language such as Mathematica. In many other cases, you will need to analyze the stability of your fixed point by using numerical methods. For this, a very good tool is AUTO [13] or its derived package XPPAUT [28]. These allow you to study the stability of your system as the values of the parameters change—for instance, by drawing bifurcation diagrams—(the use of XPPAUT is nevertheless not straightforward; *see* **Note 3**).

3.6 Simulate the Model Using Numerical ODE Solvers

This step helps you visualize the dynamics of the system, even when it is far from the fixed points. The information gathered throughout the previous steps is crucial here, since numerical routines can be unstable and yield erroneous solutions. Knowing where the fixed points should be, and their stability, allows you to check your routines.

Common routines to solve ODEs are ode45 in MATLAB [29], integrate.solve_ivp in SciPy (Python) [30] and odeint::integrate in C++ [31]. They all offer the user a wide variety of numerical solvers (both implicit and explicit, with adaptive and fixed step sizes) that suit different problems. Knowing when to use a particular solver is not easy, but having them already implemented in these languages assists in the trial-and-error process. A common dilemma faced with the modeling of large systems of equations is deciding which numerical integrator to use. Here, different aspects like the efficiency, accuracy, and stability of the methods need to be carefully analyzed. A common strategy is to assume that the ODE problem is in principle non-stiff, while it can be defined as stiff if very different timescales play a role in the system. Some clever approximations may eliminate fast timescales when formulating the problem, but this is not always possible. For stiff systems, some numerical methods will force us to use an extremely small step size to find the desired solution—explicit methods—with implicit methods performing in general much better in this case [32]. You can begin by using a non-stiff solver and then, depending on the behavior of the integrator used—accuracy of the found solution, computational time, and so on—decide whether to change it or not (*see* **Note 4**).

3.7 Parameter Fitting

If experimental data is available, you should fit the model to it. When studying the response of plants to temperature, data spanning a range of temperatures will be helpful in order to make predictions, as the reaction rates will change with different temperatures, and you will need to perform separate fits for each of them. Similarly, when studying light, either a range of daylengths or light intensities will help you discern the effect of light on the problem you are studying. Ideally, a careful experimental design should be considered when developing a mathematical model.

As the number of parameters in these models is quite high, a manual trial-and-error strategy in which we try to adjust the model predictions to the experimental results is not feasible. We need to turn to optimization methods, which try to minimize the value of a so-called energy (or cost) function. Often, this function will be the sum of the squares of the differences between the experimental values and the model predictions. Typical methods to obtain parameter fits are Monte Carlo algorithms [33], where you start with an initial arbitrary set of parameters and make small modifications to those while accepting the new parameters if the energy function satisfies particular conditions. Monte Carlo methods vary widely in their algorithms of acceptance, as well as their implementations (*see* **Note 5**). Our method of choice is simulated annealing [34].

At each step, the simulated annealing algorithm evaluates some neighboring state of the current parameter state. By modifying one of the parameters slightly and comparing the energy function associated with it (E_{new}) with the one associated with the unperturbed parameter set (E_{old}), it decides whether to accept the new parameter values or not, based on the acceptance probability:

$$P_{acc} = \begin{cases} 1 \ \text{ if } E_{\text{old}}/E_{\text{new}} < 1, \\ \\ \dfrac{E_{\text{old}}}{E_{\text{new}}} T_{\text{A}} \text{ if } E_{\text{old}}/E_{\text{new}} < 1 \end{cases},$$

where T_A is the "temperature" of the annealing, which decreases with each step. The idea is to use probabilities ensuring that the system moves to lower energy states. The particularity of simulated annealing is that the probability of accepting a parameter change resulting in an increase of the energy function decreases with each annealing step, using a particular function T_A that requires to be fine-tuned through trial-and-error. We often use:

$$T_{\text{A}} = \frac{0.8}{\sqrt{1+i}}.$$

The algorithm will be repeated until either a preset maximum number of iterations has been achieved or it successfully meets an error tolerance criterion (*see* **Note 6**).

3.8 Making Predictions

Once the model is fitted to the data, you can make predictions of the behavior of the system by varying relevant parameters. If your model includes the effect of temperature, you can predict the behavior of the plant for temperatures out of the set that has been measured. Equally, with light, we were able to predict the growth of plants at 4 h of day lengths, an experimental condition that had not been measured for our system [25].

Finally, you can also assess how changing parameters will change the behavior of the system. Perhaps a fixed point becomes unstable, thus leading to completely different dynamics. Combining this last step with the previous ones will help you to understand the behavior of the system, and gain entirely novel biological insights concerning the studied functions.

4 Notes

1. ODEs are not the only modeling framework available. If you need to introduce spatial information into your model (for

instance, because you want to understand how different parts of a stem or a leaf react to different stimuli), you will need to use partial differential equations (PDEs), which we have not mentioned here. Some relevant plant modeling studies using this framework can be found in [35].

Also, it may be that stochasticity is relevant to the system. This happens when the number of molecules involved is small (and the continuum approximation does not work), and also when the response of the cells to environmental changes is not deterministic. When this is the case, one may use stochastic processes to model the system [36]. Alternatively, stochastic differential equations (SDEs), such as the Langevin eq. [37], are very common in modeling biological systems, see for instance [38, 39].

It is worth noting that the simulation and analysis techniques differ for these methods, but detailing them is outside the scope of this chapter.

2. In some cases, simplification of the model using steady-state assumptions is not straightforward. It is not until you perform numerical simulations that you realize, for instance, that some variables reach equilibrium much faster than others; you can then assume those to be constants, consequently simplifying the simulations. Also, depending on the range of some parameter values, corresponding terms in the ODEs can be neglected. This does not mean that these interactions do not exist, but rather that they are not relevant for the system under the experimental conditions being studied.

   Most models of biological systems are "sloppy" [40], meaning that some parameters are very important to the dynamics while others are not and can thus be removed. For large models, this procedure is not easy to do by hand, and automated methods making use of the geometry of the parameter-to-function manifolds [41] will help us understand the basic structure behind these models.

   Sloppiness is a consequence of not having enough data to determine the structure of the model. This means that, for a given problem, many putative models will be compatible with the data, and so the search for the "best" model is futile. We can then only aim to understand how the data restricts the kinds of models we can build.

3. It is convenient to emphasize that despite the usefulness of XPPAUT for carrying out bifurcation analysis, its handling is very demanding. Understanding how to use the program is a complex and trial-and-error process. Despite the existing documentation about this software, the practical examples are mostly reduced to prototypical and ideal cases. For real problems, it can be highly complex not only to solve a particular

error but just to understand and find out the type of error. Nevertheless, XPPAUT and its underlying software, AUTO, are considered among the best—if not the best— computational tools for this type of analysis.

4. Numerical solvers (especially in Python) often yield numerical errors, and great care has to be taken with the parameters of the algorithm: step size, solver, and stopping conditions. Given that this process can be time-consuming, having approximate analytical forms of the solutions or fixed points, regardless of their roughness, can prove helpful. You do not want to report numerical artifacts as the solution.
5. Many Monte Carlo methods using Bayesian frameworks exist for parameter fitting, and most of them are implemented in libraries in Python (for instance, the pyABC library) and Matlab (the Markov Chain Monte Carlo Toolbox), making this step easier. However, in our experience, these libraries do not work well with the kind of problems we have worked with, perhaps because of our own limited experience with them. Therefore, we have preferred to code our own optimization algorithms. Particularly, simulated annealing has worked very well for us in many cases.
6. Model building can be carried out in a more sophisticated manner. Moreover, additional steps can be considered, such as studying the identifiability of parameters. We refer the reader to reference (42) for more details on this approach.

Acknowledgments

Research has been supported by Spanish MCIN/AEI/10.13039/501100011033/ and FEDER, EU (grant nos. PGC2018-098186-B-100 and PID2022-142185NB-C22 to P.C., PID2022-142185NB-C21 to S.A., and BIO2017-90056-R/PID2020-119758RB-I00 to S.P.), besides grant BADS, no. PID2019-109320GB-100, to S.A. and P.C. The CNB and CRAG Institutes also received the “Severo Ochoa” Centers of Excellence SEV 2017-0712 (CNB) and CEX2019-000902-S (CRAG) awards from the Spanish Ministerio de Ciencia e Innovación. We thank Jorge J. Casal for enlightening discussions.

References

1. Legris M, Nieto C, Sellaro R, Prat S, Casal JJ (2017) Perception and signalling of light and temperature cues in plants. Plant J 90:683–697
2. Ingalls BP (2013) Mathematical modeling in systems biology: an introduction. MIT press
3. Rausenberger J, Hussong A, Kircher S, Kirchenbauer D, Timmer J, Nagy F, Schafer E, Fleck C (2010) An integrative model for phytochrome B mediated photomorphogenesis: from protein dynamics to physiology. PLoS One 5:e10721

4. Feinberg M (2019) Applied mathematical sciences: foundations of chemical reaction network theory. Springer, New York
5. Pay ML, Kim DW, Somers DE, Kim JK, Foo M (2022) Modeling of plant circadian clock for characterizing hypocotyl growth under different light quality conditions. silico Plants 4: diac001
6. Avello PA, Davis SJ, Ronald J, Pitchford JW (2019) Heat the clock: entrainment and compensation in Arabidopsis circadian rhythms. J Circadian Rhythms 17:5
7. Reed MC (2004) Why is mathematical biology so hard. Not Am Math Soc 51:338–342
8. Weiße AY, Oyarzún DA, Danos V, Swain PS (2015) Mechanistic links between cellular trade-offs, gene expression, and growth. Proc Natl Acad Sci USA 112:E1038–E1047
9. Milo R, Phillips R (2015) Cell biology by the numbers. Garland Science, New York
10. C++ language. https://cplusplus.com/doc/tutorial/. Accessed July 7 2023
11. Mathworks (2023) Learn with Matlab tutorials. https://mathworks.com/support/learn-with-matlab-tutorials.html. Accessed July 7 2023
12. The Python Software Foundation (2023) The Python Tutorial. https://docs.python.org/3/tutorial/. Accessed July 7 2023
13. Doedel EJ, Oldeman B (1998) Auto-07p: continuation and bifurcation software. Concordia University Canada, Montreal, QC
14. Paik I, Huq E (2019) Plant photoreceptors: multi-functional sensory proteins and their signaling networks. Semin Cell and Dev Biol 92: 114–121
15. Sanchez SE, Kay SA (2016) The plant circadian clock: from a simple timekeeper to a complex developmental manager. Cold Spring Harb Perspect Biol 8:a027748
16. Johansson H, Jones HJ, Foreman J, Hemsted JR, Stewart K, Grima R, Halliday KJ (2014) Arabidopsis cell expansion is controlled by a photothermal switch. Nat Commun 5:4848
17. Legris M, Klose C, Burgie ES, Rojas CCR, Neme M, Hiltbrunner A, Wigge PA, Schäfer E, Vierstra RD, Casal JJ (2016) Phytochrome B integrates light and temperature signals in Arabidopsis. Science 354:897–900
18. Jung JH, Domijan M, Klose C, Biswas S, Ezer D, Gao M, Khattak AK, Box MS, Charoensawan V, Cortijo S, Kumar M (2016) Phytochromes function as thermosensors in Arabidopsis. Science 354:886–889
19. De Caluwé J, Xiao Q, Hermans C, Verbruggen N, Leloup JC, Gonze D (2016) A compact model for the complex plant circadian clock. Front Plant Sci 7:74
20. Yu JW, Rubio V, Lee NY, Bai S, Lee SY, Kim SS et al (2008) COP1 and ELF3 control circadian function and photoperiodic flowering by regulating GI stability. Mol Cell 32:617–630
21. Jang IC, Henriques R, Seo HS, Nagatani A, Chua NH (2010) Arabidopsis phytochrome interacting factor proteins promote phytochrome B polyubiquitination by COP1 E3 ligase in the nucleus. Plant Cell 22:2370–2383
22. Park E, Kim Y, Choi G (2018) Phytochrome B requires PIF degradation and sequestration to induce light responses across a wide range of light conditions. Plant Cell 30:1277–1292
23. Nieto C, López-Salmerón V, Davière JM, Prat S (2015) ELF3-PIF4 interaction regulates plant growth independently of the evening complex. Curr Biol 25:187–193
24. Osterlund MT, Wei N, Deng XW (2000) The roles of photoreceptor systems and the COP1-targeted destabilization of HY5 in light control of Arabidopsis seedling development. Plant Physiol 124:1520–1524
25. Nieto C, Catalán P, Luengo LM, Legris M, López-Salmerón V, Davière JM, Casal JJ, Ares S, Prat S (2022) COP1 dynamics integrate conflicting seasonal light and thermal cues in the control of Arabidopsis elongation. Sci Adv 8:eabp8412
26. Bender CM, Orszag SA (1999) Advanced mathematical methods for scientists and engineers I: asymptotic methods and perturbation theory. Springer, New York
27. Strogatz SH (2018) Nonlinear dynamics and chaos with student solutions manual: with applications to physics, biology, chemistry, and engineering. CRC Press, Boca Raton
28. Ermentrout B, Mahajan A (2003) Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students. Appl Mech Rev 56:B53
29. Mathworks (2023) ode45: Solve nonstiff differential equations — medium order method. https://mathworks.com/help/matlab/ref/ode45.html. Accessed July 7 2023
30. SciPy documentation (2023) scipy.integrate.solve_ivp. https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html. Accessed July 7 2023
31. Boost C++ Libraries (2023) Odeint. https://www.boost.org/doc/libs/1_74_0/libs/numeric/odeint/doc/html/index.html. Accessed July 7 2023
32. Spijker MN (1996) Stiffness in numerical initial-value problems. J Comput Appl Math 72:393–406

33. Liu JS, Liu JS (2001) Monte Carlo strategies in scientific computing, vol 75. Springer, New York
34. Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
35. Prusinkiewicz P, Rolland-Lagan AG (2006) Modeling plant morphogenesis. Curr Opin Plant Biol 9:83–88
36. Karlin S, Taylor HM (1975) A first course in stochastic processes. Academic Press
37. Gillespie DT (2000) The chemical Langevin equation. J Chem Phys 113:297–306
38. Muñoz-García J, Ares S (2016) Formation and maintenance of nitrogen-fixing cell patterns in filamentous cyanobacteria. Proc Natl Acad Sci USA 113:6218–6223
39. Casanova-Ferrer P, Ares S, Muñoz-García J (2022) Terminal heterocyst differentiation in the Anabaena patA mutant as a result of post-transcriptional modifications and molecular leakage. PLOS Comp Biol 18:e1010359
40. Transtrum MK, Machta BB, Brown KS, Daniels BC, Myers CR, Sethna JP (2015) Perspective: sloppiness and emergent theories in physics, biology, and beyond. J Chem Phys 143:010901
41. Transtrum MK, Qiu P (2014) Model reduction by manifold boundaries. Phys Rev Lett 113:098701
42. Villaverde AF, Pathirana D, Fröhlich F, Hasenauer J, Banga JR (2022) A protocol for dynamic model calibration. Brief Bioinform 23:bbab387