Analyzing mathematical models with MATLAB: numerically integrating ordinary differential equations and simulating stochastic systems

Contents

1. Motivation

In order to get a quantitative understanding of a given biological process, it is necessary to come up with a mathematical description of the process we have in mind. Give such a mathematical model, we have to analyze it in order to gain some understanding. Using a computer, even complicated looking things like multidimensional differential equations or stochastic systems can be analyzed numerically, in a surprisingly straightforward manner. This tutorial will be all about getting a handle on mathematical models by analyzing them using MATLAB.

Boxed expressions indicate stuff for you to do.

Bridging the language barrier from mathematics to biology and *vice versa* can be hard. It's possibly the biggest obstacle to overcome, but the rewards can be staggering and make it all worth it. When spending a lot of ~~much~~ time on one side of the fence it's useful to remind ourselves that not one side trumps the other. Clearly it's all hand in hand. Don't ever get involved in trying to settle what's more important! The grand goal is clear for this kind of course which is often hailed as bringing together physics and biology: we're not aiming for inter-disciplinarity but for non-disciplinarity! This course is part of that aim and will introduce some of the basic concepts that can be used to gain insights into mathematical descriptions of biological processes. It will be a starting point to "get your hands dirty with MATLAB".

2. A "`Hello World!`" differential equation

If you have gone through the first steps of programming a computer before, you will probably have seen a "`Hello World!`" example of a program. This section will introduce an equivalent differential equation.

So let's start by addressing the following question. Under appropriate conditions an *E. coli* cell will divide every 20 minutes. If you put ten *E. coli* cells into their favorite medium with lots of nutrients and space to grow at noon, how many cells will be there just after midnight when you come back from the Captain Kidd (assuming that crowding is not an issue)?

Well, you can calculate the answer pretty much directly, but let's hold that off for a moment and try to formulate the problem as a differential equation describing the number of cells x as a function of time t. Since bacteria are self-replicating the rate of change of $x(t)$ is proportional to $x(t)$, which is captured by the following differential equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = bx \quad . \tag{1}$$

The solution to this differential equation is of course an exponential of the form

$$x(t) = x_0 \mathrm{e}^{bt} \quad , \tag{2}$$

where x_0 is the number of bacteria at time $t = 0$. But even if we did not know the analytical solution we could simply integrate the differential equation for the starting point $x_0 = 10$ and $b = \ln 2/20\ \mathrm{min}^{-1}$ until the final time point $t = 12 \times 60$ min.

Relate the birth rate b and the doubling time using (2).

Note that integrating a differential equation numerically is easy. All you need to do is to specify the parameters and initial conditions and off you go. Really complicated, non-linear equations can be as straightforward as easy ones. However, finding general properties of solutions, such as limits, minima and other parameter-dependent results is not really possible with numerical solutions. For that kind of understanding we need to gain analytical insights.

Also note that in the above we simply glanced over the question of how a microscopic, discrete and possibly stochastic (the population of cells will not divide synchronously) process can lead to macroscopic, continuous and deterministic time evolution. The relation between the two will be discussed in greater detail in the latter part of this tutorial.

So the task at hand is to write a program that integrates the differential equation numerically. Fortunately MATLAB has various very powerful ODE‡ solvers built in which we can make use of. So writing an M-file will simply consist of defining the ODE and calling the built in ODE solver to determine the result of the numerical integration. (As explained in the general introduction, make sure your working directory in which you save your M-files and the path that MATLAB searches for M-files agree!).

Write an M-file that integrates equation (1) and plot the solution $x(t)$ in linear and logarithmic scale.

If you have some previous MATLAB exp[illegible]nce, try to write the M-file from scratch, using the built-in function `ode23t`. For information about the implementation of the integration routine use the built in help menu `help ode23t`.

Otherwise you can go to your work directory and look at the file `bacterial_explosion_ode.m` which defines a function whose output is the right hand side of equation (1) and the script `script_bacterial_explosion.m` which calls the ode solver to numerically integrate the differential equation. Executing the script `script_bacterial_explosion` will result in `x` and `t` to be vectors containing the respective values x_i, t_i for $x_i = x(t_i)$ and will also generate the corresponding plots.

Redo the calculation for the case in which you accidentally left your bacteria growing for 48 h. Is your result reasonable? (Hint: the mass of an *E. coli* is approximately 10^{-1}[illegible]g, and the mass of the earth is approximately 6×10^{24}kg . . .)

2.1. *Absolute and relative tolerance*

The absolute tolerance ("`AbsTol`" in the line `options=odeset('AbsTol', 1e-6, 'RelTol', 1e-6);`) is the cutoff measure below which the computer

‡ ODE = ordinary differential equation. This means that the functions we're trying to solve for depend only on one variable, which in these examples is time.

thinks a value is zero. The computer could waste a lot of time following insignificant numbers. A way the computer deals with this is to pretend that any number smaller than a certain value is really zero (it won't always print it as zero but it still treats it as zero). The absolute tolerance is this level.

Integrate equation (1) for a negative "birth rate" $-b$ (modeling radioactive decay) and plot the result logarithmically. Have a careful look at the curve.

You will notice graph is no longer a perfectly straight line. Something funny is going on. This is because the computer considers these values zero (AbsTol was set to `1e-6`) and the log of zero is undefined (and MATLAB somehow tries to deal with that).

Change `AbsTol` to `1e-10` and rerun the code.

You should notice the graph looks fine / will get broken much later.

This is worth mentioning because if you are trying to model a biological~~ly~~ system the (molar) concentrations will often be less than `1e-6` which is the default tolerance. You will need to make the tolerance about 10 fold less than what you expect the small value you care about will be if you ~~won't~~ to avoid getting deceived. Or better still, whenever possible rescale your quantities by using units that lead to the numerical values to be of the order of one.

The `RelTol` or relative tolerance is a measure of the accuracy each point on the modeling graph. A `RelTol` of `1e-6` means each point is calculated so that its numerical error is at most one part in a million.

3. Bacterial growth curves

3.1. Differential equation

In the above example growth was unlimited, which is obviously not realistic. At some point overcrowding will lead to competition for nutrients and will limit the growth. The exponential growth will end in a saturation phase and a bacterial growth curve such as the one depicted in figure 1 can typically be observed. (Hang out with Rob Phillips in the latter part of the Physiology Course for some fun measuring bacterial growth curves!).

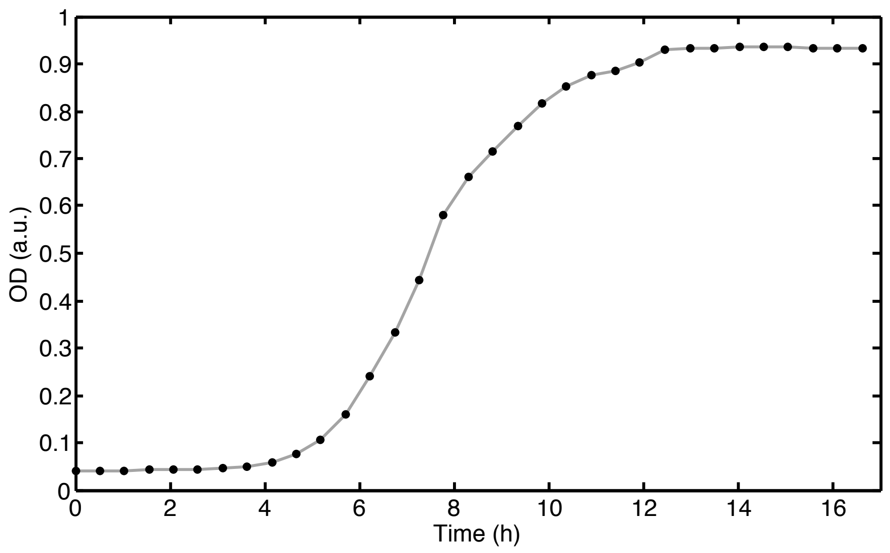

Figure 1. Typical bacterial growth curve. A growing bacterial population is measured at various time points with the optical density (OD) as a measure of cell density and therefore total number of cells.

Saturation effects are often modeled by introducing a death term that competes with the birth term kx in equation (1). For the system to saturate the death term needs to grow faster than the birth term as the number of bacteria increases. That means it has to be non-linear. The simplest ~~simplest~~ of such non-linear death terms is given by dx^2, which leads to

$$\frac{\mathrm{d}x}{\mathrm{d}t} = bx - dx^2 \quad . \tag{3}$$

Note the above equation is motivated purely phenomenologically and not by an understanding of the process of overcrowding that leads to the saturation. It is therefore not clear *a priori* whether equation (3) is a sensible description. Nevertheless, unless the form we have chosen is totally wrong it will capture some aspect of the saturation effect and will at least help us to describe and quantify bacterial growth curves.

Can you think of a possible microscopic process that leads to events happening with a rate that is proportional to x^2?

Note that most non-linear differential equations are impossible to solve analytically (this one actually is[2], which is probably why it is so popular) so numerical integration is the only way to go for many complicated systems.

Integrate equation (3) numerically for the same set of parameters as before (i.e $x_0 = 10$, $b = \ln 2/20$ min^{-1}) and $d = b \times 10^{-10}$. Plot the resulting growth curve in linear and logarithmic scale and overlay the graphs with the previous result for unlimited growth.

In case you run into problems changing the previous M-files to solve equation (1), in your work directory you can find the following two M-files to guide you: `bacterial_growth_ode.m` which defines a function whose output is the right hand side of equation (3), and `script_bacterial_growth_curve.m` which calls the ode solver to numerically integrate the differential equation.

3.2. Fixed points

In order to analyze differential equations it is often very useful to look at values of x for which the system doesn't change anymore, i.e. values of x for which the right hand side of the differential equation becomes zero. In the above example the so called fixed points of the system satisfy the following quadratic equation.

$$0 = \underbrace{bx - dx^2}_{f(x)} \quad , \tag{4}$$

which has two solutions $x = 0$ and $x = b/d$. Note that the two fixed points don't have the same property: one is stable and the other one is unstable. The difference means that if the number of bacteria deviates only slightly from zero then the whole thing will take off. Hence the time evolution of a system with exactly zero or almost zero bacteria will look extremely different. The fixed point (or steady state) $x_{ss} = 0$ is therefore called unstable. In contrast to that is the behavior of the system for a number of bacteria close to $x = b/d$, which will simply relax back into $x = b/d$. The fixed point (or steady state) $x_{\text{ss}} = b/d$ is therefore called stable.

Sketch $f(x)$ and indicate the respective regions in which $f(x)$ is positive, and where it is negative. Does that tell you anything about the stability of the fixed points?

The formal analysis of the evolution of deviations from the fixed points of a system is called linear stability analysis and leads to important insights about systems described by differential equations. It is formally done by doing a Taylor expansion of the right hand side $f(x)$ of the differential equation.

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \underbrace{bx - dx^2}_{f(x)} \quad ,$$

which we truncated after the linear terms such that,

$$f(x_{\text{ss}} + \Delta x) \approx \underbrace{f(x_{\text{ss}})}_{0} + \Delta x \, f'(x_{\text{ss}}) \quad .$$

This leads to the following linear differential equation describing the behavior of small deviations Δx from the fixed point value x_{ss}

$$\frac{\mathrm{d}\Delta x}{\mathrm{d}t} \approx \Delta x \, f'(x_{\text{ss}}) \quad . \tag{5}$$

We already know that the solution to the above equation is an exponential with $\Delta x \sim \mathrm{e}^{f'(x_{\text{ss}})t}$. So depending on the sign of $f'(x_{\text{ss}})$, a deviation is growing exponentially or decreasing exponentially. Hence the system - if nudged a little bit away from its fixed point - will either return to the fixed point or be pushed away from it. Accordingly fixed points are categorized as stable or unstable as illustrated by the physical example in the next section.

Formally show that $x_{\text{ss}} = 0$ is an unstable fixed point and that $x_{\text{ss}} = b/d$ is a stable fixed point.

Note that this kind of analysis is straightforward to extend to higher dimensional systems (i.e. systems with more than one variable), which we will do in later sections. Also note that in higher dimensional systems you can have interesting things like neutrally stable fixed points around which the system oscillates.

3.2.1. Physics analogy Imagine the motion of a ball placed in a valley where the height of the canyon walls $h(x)$ is given by $h(x) = x^4 - 40x^2$, where x is the position of the ball. Fixed points can be thought of as the positions for which a ball – if placed there – would not move. Stable fixed points are points where the ball would return if pushed slightly away from the fixed point. Unstable fixed points are points where the ball would not return if pushed slightly away from the fixed point.

Plot the function that describes the height of the valley $h(x)$. What are the fixed points of this system and are they stable?

3.3. Parameter estimation by data fitting

You'll find a whole bunch of bacterial growth curves in the work folder (`growth_curveXXX.m`). In this section we will address the following questions: can you quantify growth curves, can you estimate the confidence in the parameters you found to quantify the curves, and can you tell different growth conditions apart?

The differential equation (3) describing population growth is one of the few non-linear cases that can be solved analytically. Hence you don't have to numerically integrate the solution for a whole bunch of parameter values to see which fits the data best, but can make use of the analytic solution given by

$$x(t) = \frac{x_0 e^{bt}}{1 + (e^{bt} - 1)x_0\, d/b} \quad . \tag{6}$$

Fortunately MATLAB has some powerful algorithms built in to do the fitting for you. The one that we will use is called `fminsearch`, which allows for non-linear parameter fitting. Note that non-linear here means that the parameters enter in a non-linear way, not that the function is non-linear in t. For example fitting the function $x(t) = b * t +$ [illegible] t^2 constitutes a linear fit whereas $x(t) = d + t/d$ constitutes a non-linear fit. Since linear best fit results are much more reliable than non-linear ones, it is always advantageous to formulate your problem as a linear fit problem if possible.

Fitting a theoretical curve to data means to find a set of parameters for which the curve follows the data as closely as possible. Usually the closeness of the curve to the data is measured by summing the squared vertical residuals. Fitting then boils down to minimizing that sum of squared residuals. The result of this procedure is therefore often called a "least squares fit". Doing a least squares fit in MATLAB is relatively straightforward, the only potentially tricky part is to keep track of your parameters and the data.

For guidance have a look at the function `goodness_of_fit.m` that calculates "the distance" between the data and the curve (6). The script `best_fit_and_plot.m` minimizes that distance and returns a plot directly comparing the best fit and the raw data.

Data fitting leads to best fit parameter values which can be purely phenomenological or containing fundamental information about the system under consideration. It depends whether the fitted mathematical curve was motivated by an understanding of the underlying mechanism or not. Also note that just because a curve fits the data very well the underlying model does not need to be true! There is a famous saying attributed to the mathematician John von Neumann who is reported to have said "with four parameters I can fit an elephant, and with five I can make him wiggle his trunk". (Actually, this is not quite true, as you can see illustrated in figure 2).

The structure of the growth curve data is as follows: the data set `growth_curve_A` contains two columns, with the first column containing the time points of the OD measurements, and the second column containing the OD measurements. The data sets `growth_curve_B` and `growth_curve_C` contain fourteen columns, with the first seven columns containing the time points of the OD measurements, and the second seven columns containing the OD measurements. In order to read in data sets use the command `dlmread`, e.g. for the first growth curve, type `gcdata=dlmread('growth_curve_A.dat')`. Then you can plot the growth curve data by typing `plot(gcdata(:,1),gcdata(:,2));`.

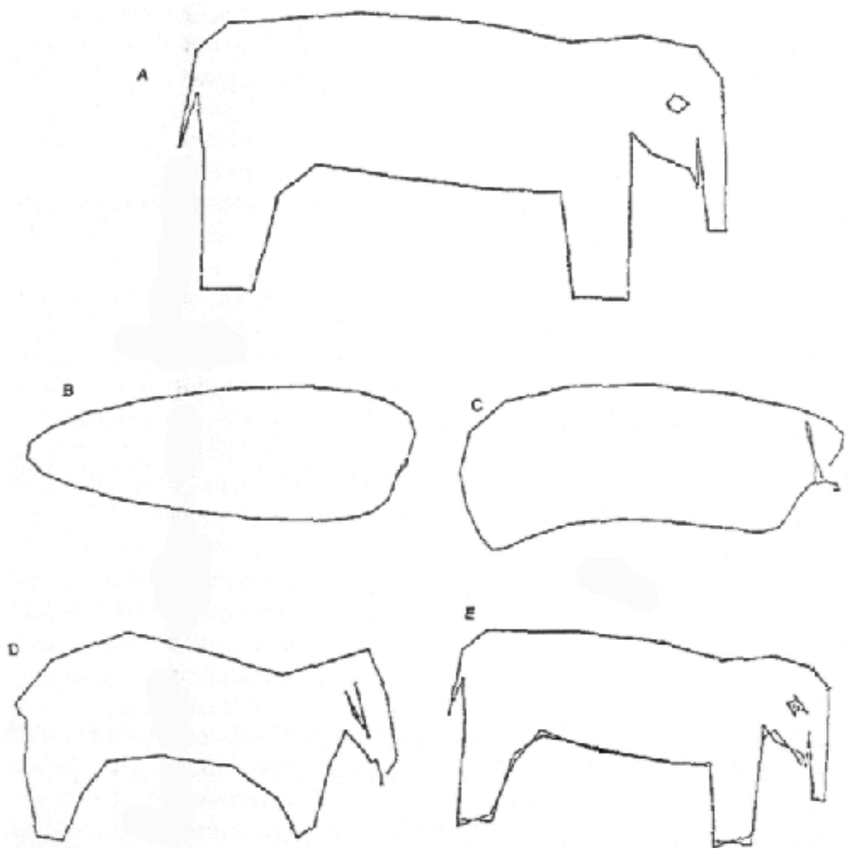

Figure 2. Mathematical biology at its best: the question "how many parameters does it take to fit an elephant?" was settled in the seminal 1975 paper by J. Wel "Least squares fitting of an elephant". (The answer is thirty).

- **Fit the solution defined by (6) to the data set `growth_curve_A` to find best fit estimates for the parameters b and d. Is there a clever graphical way to alternatively read off good estimates for those parameters from the growth curve data directly?**
- **Fit the solution to the data sets saved in `growth_curves_B` and thereby determine parameter variability for data sets obtained in identical growth conditions.**
- **Additionally fit the solution to the data sets saved in `growth_curves_C`. Can you distinguish the mutants from the WT cells in the previous data set?**

4. Polymer growth model / prion dynamics

Let's have a look at a similar example describing the dynamics between two species that includes a positive feedback loop. Biology is full of feedback loops, and as an example we will analyze a simple system with feedback in the following. Namely, a toy model of the dynamics of prion numbers of the formation of polymers

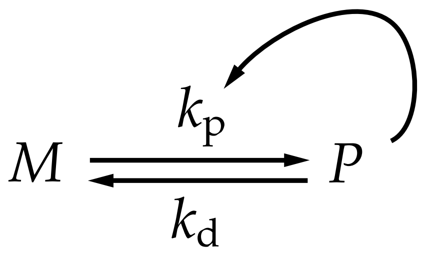

Figure 3. Illustration of the reaction scheme underlying our toy model of polymer formation. Note that this cartoon doesn't completely specify the process, because how exactly the concentration of P affects the rate of polymerization is not specified!

Let's say each molecule can be in one of two conformational states M (monomer) or P (polymer) and that M is converted to P at a rate k_p while P can break apart to form M at a rate k_d. P is assumed to enhance its own production rate (a positive feedback), and let us for simplicity say that P multiplicatively affects the rate at which M is converted to P. This reaction scheme is illustrated in figure 3.

Write out the two ordinary differential equations that describe this system (one for M and one for P). Using the conservation law that $M + P = M_{\text{tot}}$, reduce the system to one ordinary differential equation in P (involving the constants $k_{\text{p}}, k_{\text{d}}, M_{\text{tot}}$ but not M). What are the stationary state values of P and are they stable or unstable? Use the graphical method from earlier and determined the stability of the fixed points by looking at the right hand side $f(P)$ of the differential equation. Sketch by hand the stable stationary state value P_{ss} and M_{ss} (y-axis) versus M_{tot} (x-axis). Imagine M_{tot} starts at zero and is synthesized to a high level. Describe biologically what happens during the process (continue to make the assumption that the system reaches steady-state at each value of M_{tot}). This is a very simplistic model. What features of real polymers (or prions!) are we ignoring that should be included in a more complex but realistic model?

5. Translation of mRNA

Say someone asks us to make quantitative predictions about the relation between gene expression levels, mRNA levels, and protein levels in a cell. In a simple scenario, we could say that the production rate of the protein is proportional to the mRNA levels, and that mRNA is produced at a constant rate (i.e. the gene is transcribed at a constant rate), while both protein and mRNA are degraded with a constant rate per molecule. Calling the concentration of mRNA $x(t)$ and the concentration of the protein $y(t)$, these assumptions can be written in mathematical form

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= b_{\text{m}} - d_{\text{m}}x \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= b_{\text{p}}x - d_{\text{p}}y \end{aligned} \quad . \tag{7}$$

Realistic values for the production rates and degradation rates for a typical mRNA and protein pair in *E. coli* are $b_{\text{m}} = 1\ \text{min}^{-1}$, $d_{\text{m}} = 1/2\ \text{min}^{-1}$, $b_{\text{p}} = 1\ \text{min}^{-1}$, $d_{\text{m}} = 1/20\ \text{min}^{-1}$.

Let's first look at the dynamics of mRNA, which is independent of the protein levels.

Edit the original ODE solver M-file to integrate the x dynamics of the above system starting from $x_0 = 0$, i.e. find out how the mRNA and protein levels rise after the gene is suddenly switched on. Check graphically that the initial rise of mRNA levels is linear and proportional to b_m. Sketch the right hand sied of the differential equation $f(x) = b_\mathrm{m} - d_\mathrm{m}x$ and comment on the stability of the fixed point.

Next look at the full two dimensional problem in which the production of proteins depends on the instantaneous mRNA levels.

Extend the original ODE solver M-file and integrate the above system starting from $x_0 = 0$ and $y_0 = 0$, i.e. find out how the mRNA and protein levels rise after the gene is suddenly switched on. Check graphically that the initial rise of mRNA levels is linear and proportional to b_m. Comment on the time scales.

For guidance feel free to have a look at the function `mRNA_translation_ode.m` that defines the right hand side of (7) and the script `script_mRNA_translation.m` that does the numerical integration.

Many proteins in *E. coli* are degraded very slowly compared with the corresponding mRNA molecules, implying that $d_\mathrm{p} \ll d_\mathrm{m}$. This allows us to "separate time scales" in the dynamics of mRNA and proteins.

Compare the full result for protein levels with the one in which you assume mRNA levels are immediately at their steady state.

Hint: assuming that the protein dynamics is much slower than the mRNA dynamics (i.e. assuming that mRNA levels are immediately at their steady state) the system simplifies to

$$\frac{\mathrm{d}y}{\mathrm{d}t} = b_\mathrm{p}\frac{b_\mathrm{m}}{d_\mathrm{m}} - d_\mathrm{p}y \quad . \tag{8}$$

5.1. Fixed point and response time

Determine the fixed point of system (7).

In order to determine whether the fixed point is stable or not we have to linearize the system in two dimensions which involves calculating the partial derivatives of the right hand side of the differential equation. For the general two dimensional system of the form

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= f(x,y) \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= g(x,y) \end{aligned} \quad ,$$

the Jacobian matrix, which describes how small deviations from the fixed point evolve, is given by

$$\begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix}_{\mathrm{ss}}$$

where *ss* indicates that the various partial derivatives are evaluated at the stationary state fixed point. Whether a fixed point is stable or not is then determined by whether the eigenvalues of the Jacobian matrix are negative or not. Note that in multidimensional systems, fixed points can be stable in one direction but unstable in another. The directions are given by the eigenvectors of the Jacobian matrix. Eigenvalues and eigenvectors of a matrix A can be found in MATLAB by typing `[V D] = eig(A)`, which returns a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors.

Calculate the Jacobian matrix for system (7) and determine whether the fixed point of is stable by analyzing the eigenvalues of the Jacobian.

Note that there is a script in your work directory `script_mRNA_translation_jacobian.m` that you can consult for guidance.

An important property of a system is its response time, i.e. the time it takes to reach half its steady state level after starting from zero.

Looking at the simplified protein level dynamics determine its response time by numerically integrating (8). Play around with the parameters to see how you can make the system more responsive. After some numerical explorations can you "see" analytically why the parameters affect the response time the way they do?

5.2. Phase plane analysis

In order to get an understanding a system it is often instructive to visualize the the dynamics of a system in a so called phase portrait. A phase portrait (or phase plane) contains information about all possible trajectories $x(t), y(t)$. The nature of a phase portrait in which x versus y for various trajectories are depicted can be found by the linear stability analysis described earlier, which allowed you to find the stable and unstable directions in which trajectories enter or leave the fixed point. Together with the nullclines, which are the lines for which one of the derivatives vanishes, meaning that tangents to trajectories are vertical (for $\frac{dx}{dt} = 0$) or horizontal (for $\frac{dy}{dt} = 0$), this information often suffices to get a very good idea about the system's dynamics.

Plot the trajectory you $x(t), y(t)$ you determined earlier for a starting point $x(0) = y(0) = 0$ in a phase plane. Play around with other starting points and sample the phase portrait that way. Have a look at the M-files `dfield7.m` and `pplane7.m` and see whether you can use them to make a fully automated phase diagram plot.

6. Mutual inhibition

It is not uncommon for cellular components to mutually inhibit each other. Say, X inhibits the production of Y and Y inhibits the production of X, while both X and Y are being degraded independently. Assuming that there is a simple Michaelis-Menten type of inhibition this leads to the following set of

ODEs describing the dynamics of teh concentrations of x and y.

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= \frac{K_1}{K_1 + y} - d_1 x \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= \frac{K_2}{K_2 + x} - d_2 y\end{aligned} .$$

First, make a prediction how many stable states the systems has. Then determine the fixed points by sketching the nullclines for which $\frac{\mathrm{d}x}{\mathrm{d}t} = 0$ and $\frac{\mathrm{d}y}{\mathrm{d}t} = 0$. Go back to the potential well example earlier and think about the possibility of having only two steady states, both of which are stable.

The above example was a particular case of mutual inhibition. In general more complicated forms of the inhibition function can lead to multiple stable states. Interestingly, there exists a general mathematical condition for bistability that we can derive in the following.

Mathematically the general system for mutually inhibiting species can be expressed that as

$$\begin{aligned}\frac{\mathrm{d}x}{\mathrm{d}t} &= f(y) - x \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= g(x) - y\end{aligned} ,$$

where f and g are some unspecified functions that are positive, decreasing and bounded.

Depending on the exact form of the functions f and g the syestem can have multiple steady states, as illustrated in figure 4

Another, often easier, way of looking at the system is to combine the nullclines into $x = f(g(x))$. To have two stable steady states, we must have three steady states in total – two stable ones separated by an unstable. The fact that both f and g are bounded and decreasing means that there must be at least one stable steady state (see figure 4). The function $f(g(x))$ crosses the diagonal x from above for a stable steady state, and from below for an unstable steady state. Examples of systems with one or two respective stable stats are plotted in figure 5. The condition that $f(g(x))$ must cross the

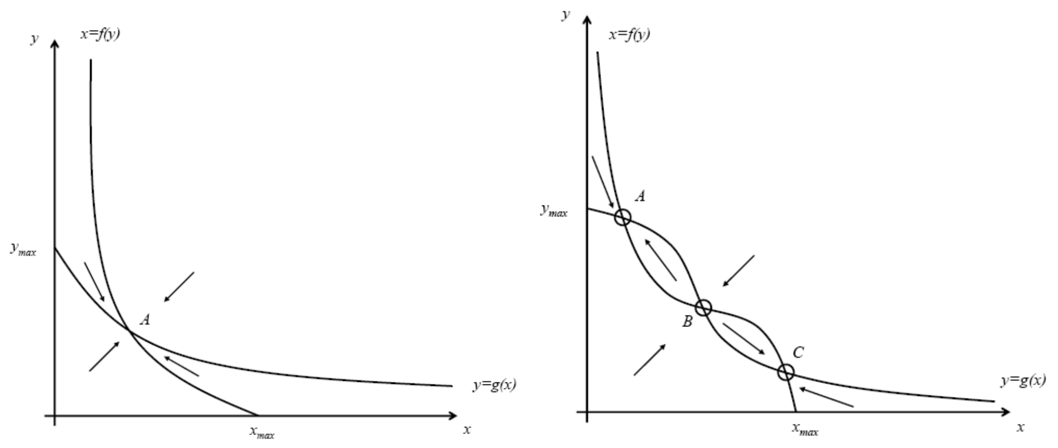

Figure 4. Mutual inhibition nullclines. Here are two specific examples for the functions f and g. Displayed are the nullclines which are given by $f(y) - x$ and $g(x) - y$. You see that in the first case, there is a single steady state, while in the second case, there are three steady states, two of which are stable. The arrows are drawn as follows: At the y-nullcline, the change in y is zero. Above the y-nullcline, y is higher and degradation will dominate over synthesis (see the rate equations) so the arrows point down, and vice versa below the nullcline. At the x-nullcline, x does not change. To the right of the x-nullcline, where x is higher, degradation of x will dominate over synthesis (see the rate equations) so the arrows point left, and vice versa to the left of the nullcline. Carrying out these arguments in each area in the graph motivates the arrows.

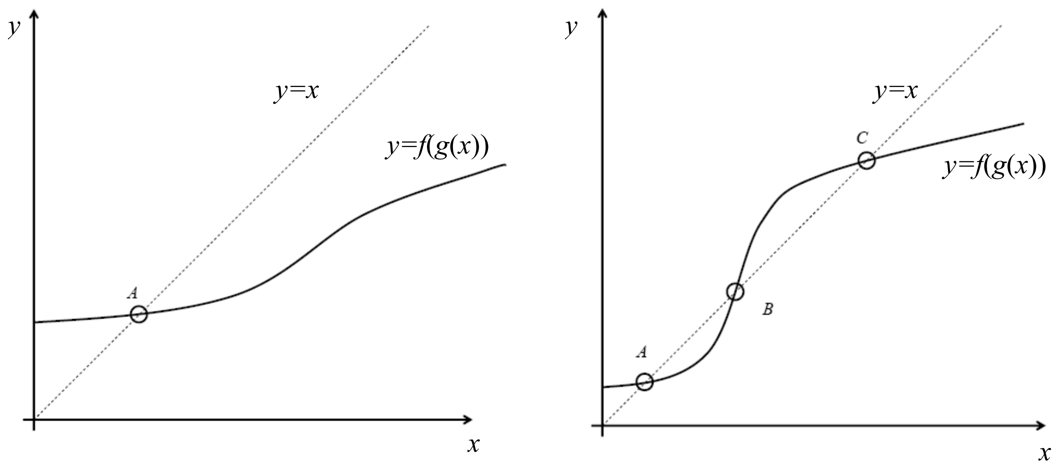

Figure 5. Mutual inhibition combined nullcline. For multiple steady states $f(g(x))$ must cross the diagonal from below, which implies that the derivative of $f(g(x))$ must be greater than the derivative of the diagonal.

diagonal from below leads to the following mathematical condition.

$$\frac{\partial f(g(x))}{\partial x} > 1 \Rightarrow \left(\frac{\partial f}{\partial y}\right)_{\text{ss}} \left(\frac{\partial g}{\partial x}\right)_{\text{ss}} > 1 \quad .$$

7. Discrete stochastic systems

The overwhelming majority of mathematical analyses of biological systems are done within the framework of differential equations. That is because the quantities of traditional physics such as position, speed, gravitational pull are all readily described as continues variables with deterministic rates of changes. The same is true for concentrations of chemicals within a well stirred reaction container. But cells are not well stirred containers with high concentrations of reagents! Genes, mRNA and proteins crucial for many cellular processes can be present in very low numbers and their individual production and degradation is the result of stochastic chemical events. This next section will take the discreteness of events and their probabilistic nature into account and formulate a mathematical framework to describe such cellular processes. Although they can be hard to solve analytically there exists a clever and very straightforward algorithm to simulate stochastic processes using MATLAB with only a handful of lines of code.

Note that clearly there is not one right way of doing things. You cannot say that "ODEs are better / worse than stochastic descriptions"! In certain circumstances and limits one mathematical description might be more appropriate than others, e.g. deterministic rate equations can follow from a limit of the average behavior of stochastic systems as we will see in the next section.

7.1. A memoryless elementary chemical reaction leads to a random walk

Consider a reaction that takes a molecule from state A to state B in a single jump. The process could for example by be radioactive decay. Instead of assuming that we have an astronomically large number of molecules in state A such that the number of molecules A in state A follows

$$\frac{\mathrm{d}A}{\mathrm{d}t} = -\lambda A$$

we want to analyze the system's dynamics in the limit of small numbers, where we cannot predict what the state of the molecule will be in the next moment, but can only determine the probability with which a molecule is still in state A.

The probability that the molecule is still in state A at time equals the probability that it was in state A at time t minus the probability $P_{\text{jump}}(t, t+\mathrm{d}t)$ that it jumped from A to B during the interval $\mathrm{d}t$

$$P(A, t+\mathrm{d}t) - P(A,t) = -P_{\text{jump}}(t, t+\mathrm{d}t) \quad .$$

Introducing the conditional probability $P_{\text{jump}}(t, t+\mathrm{d}t|A,t)$ and dividing both sides by $\mathrm{d}t$ this becomes

$$\frac{P(A, t+\mathrm{d}t) - P(A,t)}{\mathrm{d}t} = -\frac{P_{\text{jump}}(t, t+\mathrm{d}t|A,t)}{\mathrm{d}t} P(A,t) \quad .$$

which in the limit of $\mathrm{d}t \to 0$ becomes

$$\frac{\mathrm{d}P(A,t)}{\mathrm{d}t} = -\lambda P(A,t) \quad ,$$

where λ is the instantaneous transition probability per unit time. Note that in general the transition rate can depend on A. For a constant transition rate λ such as for radioactive decay, the probability that a molecule is in state A is given by

$$P(A,t) = \mathrm{e}^{-\lambda t} \quad ,$$

and the probability for the molecule to be in state B follows directly

$$P(B,t) = 1 - P(A,t) = \underbrace{1 - \mathrm{e}^{-\lambda t}}_{F(t)} \quad ,$$

where $F(t)$ is the distribution function for the reaction time. This is perhaps not obvious at first, but is both central to the following analysis and straightforward: If the system is in state B at time t, then the reaction must have occurred before time t. In other words, the probability of being in state B at time t is the probability that the reaction time was t time-units or less – which by definition is called the distribution function. This particular distribution is called the exponential distribution and the corresponding density function $f(t)$ can be found by differentiation:

$$f(t) = \frac{F(t)}{\mathrm{d}t} = \lambda \mathrm{e}^{-\lambda t}. \tag{9}$$

The exponential distribution is central to everything we will do in the "stochastic section" because it describes the randomness associated with an individual elementary chemical event. It is fairly straightforward to show that the average time $\langle t \rangle$ and standard deviation σ_t of the exponential distribution are given by

$$\langle t \rangle = \sigma_t = \frac{1}{\lambda} \tag{10}$$

This means that the standard deviation in the waiting time at for each reaction is as large as the average waiting time, which makes the overall process rather erratic.

Plot the exponential distribution (9). "For extra credit" derive its average and standard deviation as given in (10).

8. Bacterial growth

When introducing equation (1) for exponential growth of bacteria we have followed good tradition and simply glanced over the fact that the motivation for exponential growth was given on a microscopic rather than population level. Assuming that cells divide every twenty minutes, but are randomized in their timing, it is not immediately clear that for a large number of cells the population should satisfy equation (2). So let's derive the macroscopic equation (2) from the underlying microscopic process. This will illustrate the connection between deterministic rate equations and population averages of stochastic systems.

8.1. Chemical master equation

As earlier in the memoryless, elementary reaction we can only discuss the time evolution of probabilities and no longer the number of cells. Defining $P(x)$ as the probability of the systems to be in state x, i.e. that there are x cells, we can formulate an equation for the time evolution of the probabilities $P(x)$. Note that's really an infinite number of equations as $x = 0, 1, 2, , \ldots$ The birth of a cell leads to a transition of the system from state s to the state $x + 1$, and the birth rate in a population of x cells is simply bx, which we summarize diagrammatically,

$$x \xrightarrow{bx} x+1$$

That means that the probability $P(x)$ can change in two ways: the system can enter state x from state $x-1$ if a birth event happened in state $x-1$, or the system can leave state x if a birth event happened in state x. Note that these are the only two transitions that can directly affect the probability of the system to be in state x. The rate of change of the probability $P(x)$ is therefore given by

$$\frac{\mathrm{d}P(x)}{\mathrm{d}t} = b(x-1)P(x-1) - bxP(x) \quad . \tag{11}$$

This equation is called the chemical master equation (CME) because all quantities of interest can in principle deduced from it. So let's try to understand how the population average $\langle x\rangle$ behaves. Multiplying (11) with x and summing over all x leads to

$$\frac{\mathrm{d}\langle x\rangle}{\mathrm{d}t} = b\sum_{x=0}^{\infty} x(x-1)P(x-1) - b\sum_{x=0}^{\infty} x^2P(x) \quad , \tag{12}$$

(Recall that x is just an index, the variables whose time evolution we are trying to derive are the probabilities $P(x)$.)

Before proceeding convince yourself that

$$\sum_x x\frac{\mathrm{d}P(x)}{\mathrm{d}t} = \frac{\mathrm{d}\langle x\rangle}{\mathrm{d}t}$$

The right hand side of (12) simplifies after shifting the index in the first sum and making use of the fact that $P(x) = 0$ for all $x \le 0$, as there cannot be a negative number of cells, to

$$\begin{aligned}\frac{\mathrm{d}\langle x\rangle}{\mathrm{d}t} &= b\sum_{x=0}^{\infty}(x+1)xP(x) - b\sum_{x=0}^{\infty} x^2P(x)\\ &= b\sum_{x=0}^{\infty} xP(x)\\ &= b\langle x\rangle \quad .\end{aligned}$$

Hence we see that if we take many populations of stochastically dividing cells, the average over all populations (or the average over many reruns of the same experiment) indeed follows the deterministic rate equation introduced in (1). Note that the variability riance of the system

8.2. Exact simulation

Since in any given state of the system there is only one "reaction" that can happen, all we need to do to simulate the stochastic process is to decide when those reactions happen. But we already know how to do that! For a given rate λ we showed that the waiting time until the next reaction happens is exponentially distributed according to (9). So all we need to do is to draw a sequence of random waiting times from the appropriate distribution. The only (very) slight complication is that the reaction rate bx depends on the state x of the system, which means that the distribution from which we have to draw the random waiting times will be different from state to state. So in order to simulate the stochastic replication of bacteria, starting with x cells, we simply have to do the following steps

(i) The system is in state x at time t, which implies that the reaction rate is given by bx

(ii) Draw a random reaction time Δt according to the exponential distribution $bx\mathrm{e}^{-bx\Delta t}$

(iii) Update the state of the system from x to $x+1$ at time $t+\Delta t$. Go back to step (i) and repeat *ad nauseam* ...

The only practical issue we might have to think about, is how to draw a random reaction time from the appropriate distribution, given that MATLAB is happy to give us a uniformly distributed random number between zero and one. In order to get an appropriately random number we have to invert the cumulative distribution function $F(t) = 1 - \mathrm{e}^{\lambda\Delta t}$. Starting from a uniform random number u between zero and one (as given by MATLAB), we get an appropriately distributed Δt, by calculating

$$\Delta t = -\frac{1-u}{\lambda} \quad .$$

If the above is not entirely clear, convince yourself that by graphically examining what it means to invert $F(t)$, and by thinking about how to calculate probabilities for t to lie in a certain interval.

Write an M-file that iterates steps (i)-(iii) and simulates the bacterial explosion system, starting from $x = 10$ until $x = 100, 1000, 10^5$. Plot the time trace of $x(t)$. Also play around with different initial values $x(0)$ and comment on the difference between stochastic and deterministic time traces and how that difference depends on the number of cells. By running a whole bunch of simulations verify that the average of the stochastic simulations satisfies $\langle x \rangle \sim e^{bt}$.

If you get stuck, have a look at the script `bacterial_explosion_stoch_sim.m` which runs the simulation and returns a vector `t` with the transition times, which allows you to plot $x(t)$. In order to plot the data appropriately, use `stairs(t,x)` instead of the usual plot command, so the data points are not interpolated, and the discrete steps remain discrete. Note that the above simulation scheme is a special case of the famous Gillespie algorithm, that is used to simulate stochastic systems. In the following sections we will study systems that can undergo more than one reaction (in life there is death as well as birth . . .) for a given state and thereby introduce the full Gillespie algorithm.

9. Translation of mRNA

For simplicity look at the dynamics of mRNA translation in which the dynamics of mRNA is fast compared with the protein dynamics, i.e. the protein levels as defined by equation (8). Calling for simplicity $\bar{b}_p = b_p b_m / d_m$, we can write the dynamics of protein production and degradation as a birth and death process with respective birth and death rates $\bar{b}_p$,d_p. Schematically we represent this birth and death process in which the number of proteins y changes probabilistically as follows

$$y \xrightarrow{\bar{b}_\mathrm{p}} y+1$$
$$y \xrightarrow{d_\mathrm{p} y} y-1$$

Note that the "rates" $\bar{b}_\mathrm{p}$ and d_p are now transition probabilities per unit time rather than deterministic rates.

In any given state the above system can undergo one of two possible reactions: a death event could occur or a birth event could occur. We have to extend the previous algorithm to account for that fact. The way that can be done is straightforward. The reaction time will now be distributed exponentially with a total reaction rate r_T, which is the sum of all reactions rate, i.e. $r_\mathrm{T} = \bar{b} + d_\mathrm{p} y$. That will then be the appropriate random reaction time for a system in state y to undergo a transition. Note that just determines that **a** reaction has occurred at that time, but not **which** reaction! In order to determine that we simply have to look at the relative rates between the events to pick the possible reaction events with the correct probability. In this case, there are only two reactions which means that given an event as occurred after the given reaction time, a birth event has occurred with probability $\bar{b}/r_\mathrm{T}$ and a death event has occurred with probability d/r_T. Note that the two probabilities add up to one as they should (one and only one event as occurred). So the more general simulation scheme is given by

(i) The system is in state y at time t, which implies that the birth rate is given by $\bar{b}$, the death rate is given by dy and the total reaction rate is given by $r_\mathrm{T} = \bar{b} + d_\mathrm{p} y$.

(ii) Draw a random reaction time Δt according to the exponential distribution $r_\mathrm{T} \mathrm{e}^{-r_\mathrm{T} \Delta t}$

(iii) Pick whether a birth or death event has occurred at Δt with appropriate relative probabilities. I.e. at time $t + \Delta t$ update the system from y to $y+1$ with probability $\bar{b}/r_\mathrm{T}$ (a birth event has occurred) / update the system from y to $y-1$ with probability $1 - \bar{b}/r_\mathrm{T}$ (a death event has occurred). Go back to step (i) and repeat *ad nauseam* . . .

The above can be generalized to simulate an arbitrary number of species undergoing any number of reactions. That general simulation, which is still reproducing the exact stochastic system as defined by an arbitrary CME, is called the Gillespie algorithm.

Modify the previous M-file to this stochastic system and use it to simulate the system starting from $y = 0$ molecules until it has reached its "stationary state". Play around with the total number of simulated reactions from just a few to a lot. Compare the resulting stochastic time series with the deterministic result earlier. Simulate a bunch of stochastic traces for the system starting at $y = 0$ and determine the average behavior. Also determine the variability between the different simulations. Would you expect to pick up such variability in an experiment? Comment on what is and what is not constant for the stochastic system at its "stationary state".

For guidance have a look at `mRNA_translation_approx_stoch_sim.m`. Looking into the possibility that proteins are produced in bursts we introduce the following stochastic process

$$y \xrightarrow{\bar{b}_\mathrm{p}/10} y + 10$$
$$y \xrightarrow{d_\mathrm{p} y} y - 1$$

Do the same as before and comment on the similarities and differences. What is the variability of the protein level now? Would you expect to pick it up in an experiment?

(The corresponding script is `mRNA_translation_approx_stoch_sim_burst.m`)

10. The prion model – a nonlinear system with feedback

Going back to the toy model of prion dynamics, it will be very instructive to compare the solution of the deterministic ODE

$$\frac{\mathrm{d}P}{\mathrm{d}t} = \underbrace{(k_\mathrm{p} M_\mathrm{tot} - k_\mathrm{d})}_{k_\mathrm{d,eff}} P - k_\mathrm{p} P^2 \quad ,$$

with the dynamics of the stochastic system

$$P \xrightarrow{k_{\mathrm{d,eff}}P} P+1$$
$$P \xrightarrow{k_{\mathrm{p}}P^2} P-1$$

Note that in contrast to the previous stochastic systems we simulated, this one has transition rates that depend nonlinearly on the state variable.

Taking the following parameter values $k_{\mathrm{d,eff}} = 40$, $k_{\mathrm{p}} = 2$, integrate the deterministic ODE and compare it with the average dynamics of the stochastic system. Do you notice anything weird? Is there a massive take home message here?

11. Microtubule dynamics

One of the most spectacular processes in biology is the process of cell division. During mitosis microtubules catch the copied chromosome and pull them towards the two opposing centrosomes. For that purpose microtubules undergo a process called dynamic instability, in which they are randomly switching between a growing and a shrinking state. During mitosis that allows microtubules to find the chromosomes which they can then attach to and pull towards the respective centrosome. (Hang out in the vicinity of Tim Mitchison for the reminder of the course if you want to know more about microtubule dynamics and mitosis. . .) In order to simulate dynamically unstable microtubules you can simply write down the rate equations for switching between the two states, and let the microtubules grow or shrink deterministically in between switching events. For a given state there is only one reaction that can happen (shrinking to growing or growing to shrinking, respectively), hence you can make good use of the previous code for the stochastic simulation of the bacterial growth explosion. The only "complication" is to take into account that microtubules cannot shrink below zero length. Simply assume that when a microtubule has shrunk to zero it will switch into a growing state (this may seem weird but experimentally this corresponds to simply switching to new growing microtubule after the one you have followed has disappeared).

Simulate the dynamics of microtubule growing with 8.6 μm/min **and shrinking with** 12.2 μm/min**, given the following switching rates: from growing to shrinking (catastrophe rate)** $f_{\text{cat}} = 3.3\ \text{min}^{-1}$ **from shrinking to growing (rescue rate)** $f_{\text{cat}} = 0.27\ \text{min}^{-1}$**. Plot example trajectories for the microtubule length** L **with and without taking care of the fact that microtubules cannot have negative length. Does it make sense that in the latter case the microtubules shrink indefinitely on average? What is the average microtubule length?**

For guidance look at the script `dynamic_instability_stoch_sim.m`. Note that you could now decide to analyze what the dynamics of dynamically unstable microtubules looks like, taking into account that the process of growing and shrinking itself is stochastic. This leads you to stochastic processes whose rates are stochastic variables themselves. . . and with that you have pretty much arrived at the forefront of ongoing research in this field. Congratulations!