1.4 Using computers to solve differential equations

We have been looking so far at differential equations whose solutions can be constructed from "elementary functions," functions that we can write down in some simple form, look at and (hopefully) understand. In general, this isn't possible and in fact it might just be a historical artifact that certain functions have names and others don't. If you think you have the right equation to describe a system you are interested in, the fact that you can't immediately write down the solution shouldn't stop you. You can make approximations, which generates a lot of intuition, and you can use a computer to generate numerical solutions. This latter approach is general, and you should learn how to do it to the point where you feel comfortable.

Let us start with the simplest equation for first order chemical kinetics, in which some molecule A is tranformed into B with a rate constant k. The concentration c_A of A molecules obeys the equation

$$\frac{dc_A}{dt} = -kc_A. \tag{1.148}$$

As you all know by now, the solution is $c_A(t) = c_A(0)\exp(-kt)$. Let's see how we could find this solution numerically, check against the analytic solution to see that our strategy works, and finally use the same strategy to look at equations that are not so easy to solve with pen and paper.

Recall that the derivative is defined in calculus as the limit of finite differences:

$$\frac{dc_A(t)}{dt} \equiv \lim_{\Delta t \to 0} \frac{c_A(t+\Delta t) - c_A(t)}{\Delta t}. \tag{1.149}$$

The key to numerical solutions of differential equations is in essence to take a giant step backward and work with a finite value of Δt, hoping that we can make it small enough that we start to see the limiting behavior. In the simplest case we just make the replacement

$$\frac{dc_A}{dt} \to \frac{c_A(t+\Delta t) - c_A(t)}{\Delta t} \tag{1.150}$$

in the differential equation, and proceed:

$$\frac{dc_A}{dt} \to \frac{c_A(t+\Delta t) - c_A(t)}{\Delta t} = -kc_A(t) \tag{1.151}$$

$$c_A(t+\Delta t) - c_A(t) = -[k\Delta t]c_A(t) \tag{1.152}$$

$$c_A(t+\Delta t) = [1 - k\Delta t]c_A(t). \tag{1.153}$$

If we decide to measure time in discrete ticks of a clock, where the time between ticks is Δt, then every time $t = n \cdot \Delta t$, where $n = 0, 1, 2, 3, \ldots$. Thus instead of writing $c_A(t)$, we can write $c_A(n)$, and of course $c_A(t+\Delta t) = c_A(n+1)$. This means that Eq (1.153) really is an equation that generates $c_A(n+1)$ from knowledge of $c_A(n)$:

$$c_A(n+1) = [1 - k\Delta t]c_A(n). \tag{1.154}$$

If we start with some value of $c_A(0)$, Eq (1.154) tells us how to generate $c_A(1)$, and then we can use this iteratively to generate values for c_A at all discrete times n. In effect this allows us to "walk" through time, updating the value of c_A based on the previous value, and in this way we generate a "numerical solution" to our differential equations.

Let's see how this works in MATLAB. We'll choose units where the rate constant $k = 1$, and our "small steps of time" will be $\Delta t = 0.01$; we'll have to come back to the question of whether this choice of Δt is a good one. We'll explore times starting at $t = 0$ and ending at $t = 5$ (again, in units where $k = 1$), which means that we need to run for 500 ticks of our discrete clock. With these remarks in mind, the program becomes

```
cA = zeros(500,1);
cA(1) = 1;
k = 1;
dt = 0.01
for n=2:length(cA);
    cA(n) = (1-k*dt)*cA(n-1);
end;
```

Notice that we start by setting aside space for the thing we trying to compute, and we have to set (in the second line) its initial value. A peculiarity of MATLAB is that you start counting at `n=1`, not at `n=0`. We also have lines which define the value of the rate constant `k` and time step Δt, which is symbolized by `dt` in the program; again, our choices of these parameters are just for illustration at the moment. Once you run the program you have stored the "data" on concentration as a function of time, and you'd like to plot it. If you want things in physical units it's convenient to make a real time axis,

```
timeaxis = dt*[0:499];
```

where we are careful to note that the time corresponding to `n=1` is actually $t = 0$. Then you can type

```
figure(1)
plot(timeaxis,cA)
xlabel('time (seconds)')
ylabel('concentration of A')
```

and you should get a reasonable plot with properly labeled axes, and this will appear on your screen in a box marked Figure 1. This is not the place for aesthetic hints, but at some point you'll want to learn how to make things look nice—what is important here is to be sure that when you look at the plot you can read the units! The results are shown in Fig 1.15, where we compare the numerical solution to a numerical evaluation of the analytical result. In this simple case, it's clear that our numerical strategy "works," in that it gives us a solution that agrees with the exact mathematics.

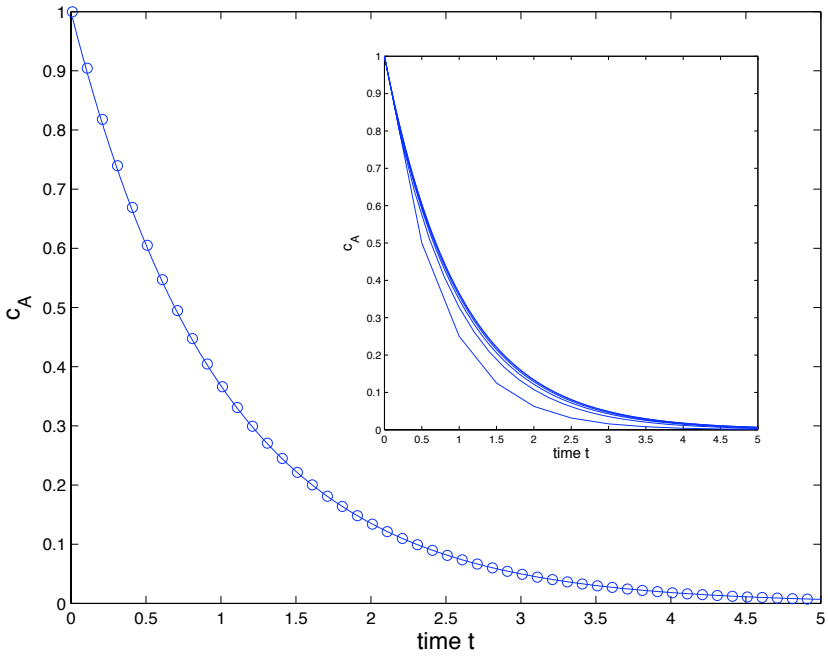

Figure 1.15: Numerical solution of the simple differential equation, Eq (1.148). As described in the text we use the algorithm defined by Eq (1.154), with the rate constant $k = 1\,\mathrm{s}^{-1}$ and the initial condition $c_A(1) = 1$ (in some units). The solid line is the result of 500 iterations with $\Delta t = 0.01\,\mathrm{s}$, and the circles show the exact solution over the same time window. Inset shows what happens as we increase Δt, with separate curves for $\Delta t = 0.01$, 0.02, 0.05, 0.1, 0.2, 0.5.

We have of course chosen here a VERY simple example. In particular we have an exact analytical solution, so using the computer is just for the sake of learning how to do it. In the problems we ask you to play a bit, changing parameters to see what is essential in making things work. In the approach suggested here, the basic issue is how to choose the size of the discrete time step `dt`. The whole idea behind our scheme is to replace derivatives with differences, and this gets to be a bad approximation if we make our steps too big, as you can see in the inset of Fig 1.15. So this pushes us toward smaller and smaller values of `dt`. But if we take very small steps then we need to take lots of steps to cover the same amount of real time, and so our computation becomes inefficient.

Thanks to the increasing speed of the devices in your computer's chips, running programs for many time steps is less of a problem than it used to be, but still there are many situations in which one will need to push the tradeoff between accuracy and efficiency. To do this, one needs to understand something about the problem you are solving, but one can also try to use more intelligent mappings from the continuous differential equation down to the discrete time steps. This is a whole field of research (numerical analysis), and as time permits we'll give you glimpses. For now, it would be good if you felt comfortable with the simplest approaches, so that faced with some new differential equation you don't how to solve, you can go to the computer and quickly see what the solutions look like—and have ways of testing to see if you believe what the computer is telling you!

Problem 18: The basic idea of this section has been to take the differential equation

$$\frac{dc(t)}{dt} = -kc(t), \tag{1.155}$$

replace it with a discrete equation

$$\begin{aligned} \frac{c(t+\Delta t) - c(t)}{\Delta t} &= -kc(t) && (1.156)\\ \Rightarrow c(t+\Delta t) &= (1 - k\Delta t)c(t), && (1.157) \end{aligned}$$

and then turn this rule directly into an algorithm.

(a.) Consider the case where $k = 10\,\mathrm{s}^{-1}$. Try various values for Δt (e.g., $\Delta t = 0.001,\ 0.01,\ 0.1,\ 1\,\mathrm{s}$) and run your program for a number of iterations that corresponds to one second of real time. Compare your numerical results with the analytic solution $c_A(t) = c_A(0)\exp(-kt)$. How small does Δt need to be in order to get the right answer? How would your answer change if the rate k were ten times faster?

(b.) Write the analogous program for a second order reaction, $A + B \stackrel{k_2}{\rightarrow} C$, described by the differential equations

$$\frac{dc_A}{dt} = -k_2 c_A c_B \tag{1.158}$$

$$\frac{dc_B}{dt} = -k_2 c_A c_B. \tag{1.159}$$

(c.) Assume initial concentrations of $c_A(0) = 1\,\mathrm{mM}$ and $c_B(0) = 2\,\mathrm{mM}$. Let $k_2 = 10^6\,\mathrm{M}^{-1}\mathrm{s}^{-1}$. Before you run your program, what value of Δt seems reasonable? For how long (in real time) will you need to run in order to see most of the interesting dynamics?

(d.) Run your program using the parameter settings from part [c]. Is there an analytic solution to which you can compare your results? If you don't have such a solution, how do you decide whether your program is giving the right answer?

Problem 19: Let's try to use these ideas to solve the equations for motion under the influence of gravity. Going back to the discussion in Section 1.1, if the height of a particle with mass m is given by $h(t)$, then Netwon's equation becomes

$$m\frac{d^2h}{dt^2} = -mg, \tag{1.160}$$

again in the limit where we take the force of gravity to be constant. Since we have discussed ways of solving equations with one derivative, but not two derivatives, let's rewrite this as two equations:

$$\frac{dh}{dt} = v, \tag{1.161}$$

$$\frac{dv}{dt} = -g. \tag{1.162}$$

Notice that units are arbitrary. Suppose that we define variables $\tilde{h} = h/h_0$, $\tilde{t} = t/t_0$, and $\tilde{v} = v/v_0$, where we choose the velocity scale to be the initial velocity, $v_0 = v(0)$.

(a.) Write the differential equations for these new variables, that is

$$\frac{d\tilde{h}}{d\tilde{t}} = \cdots, \tag{1.163}$$

$$\frac{d\tilde{v}}{d\tilde{t}} = \cdots. \tag{1.164}$$

(b.) Show that by choosing the scales h_0 and t_0 correctly, you can make even the constant g disappear from the equations. What does this mean, qualitatively, about the form of the solutions to these equations?

(c.) Write a program to solve these "dimensionless" equations, discretizing into time steps of size Δt as before. Run the program, and compare your results with the exact solution from the discussion in Section 1.1.