Chapter 10

10. DIFFERENTIAL EQUATIONS: PHASE SPACE, NUMERICAL SOLUTIONS

Abstract

Solving differential equations analytically is not always the easiest strategy or even possible. In these cases one may use a numerical solution to characterize the dynamics governed by the ODE. A diagram of the flow plotted versus time can create insight into the dynamical behavior governed by the ODE. In addition, the so-called phase space representation where the dynamics is represented by a vector at each state of the system is a helpful tool to qualitatively assess system dynamics.
Here, we examine both numerical solutions and graphical representations of the dynamics using two growth models: one model with unrestricted growth (a linear ODE), and one where the growth is restricted by limited resources (the logistic equation, a nonlinear ODE). We use different techniques and MATLAB examples to demonstrate the numerical approach: Euler's method, the improved Euler's method, and the 4^{th} order Runge-Kutta algorithm. In the final part we briefly discuss methods for obtaining solutions for partial differential equations (PDEs).

Keywords

Phase space
Euler's Method
Runge-Kutta Method
Nonlinear ODE
Logistic Equation
Partial Differential Equation (PDE)
von Neumann Criterion

10.1 Graphical Representation of Flow and Phase Space

In the previous Chapter we explored ODEs and how to solve them. In the examples we discussed in that Chapter we were able to obtain solutions in a relatively straightforward manner. There are many types of ODEs however where such a solution is not easily obtained, or using an analytical approach may even be impossible. The only approach in such a case is to compute a solution that satisfies the dynamics governed by the ODE. Especially in the lower order cases, a good start to obtain insight into the dynamics determined by the ODE is to depict the flow of the dynamics graphically. If we deal with a first order case, where the dynamics of variable S depends only on itself (i.e. $\dot{S} = S$), we can depict the direction of the flow $\dot{S}$ against the time axis (Fig. 10.1). The short lines in Figure 10.1 show how the dynamics flow. Here it can be seen that there is an infinite amount of solutions that would satisfy the dynamics.

Fig. 10.1

The following MATLAB script, pr10_1.m was used to create Figure 10.1

```
% pr10_1.m
% Solver Demo
% for dS/dt=kS

clear
close all
k=1;
Dt=.1;
figure;hold
for t=-3:.5:3;
   for S=-5:.25:10;
      DS=Dt*k*S;
      line([t-Dt t+Dt],[S-DS S+DS])
   end;
end;
```

In order to determine a single solution amongst all the possible ones, one needs to know at least one position. For example if we know that $S = 1$ at $t = 0$, the initial condition indicated by the asterix in Figure 10.1, we can determine the solution that passes through that condition (green-red line). This shows graphically why expressions such as the ones in Equations (10.6), (10.11a, b) that solve an ODE, require some boundary condition.

Another common procedure to explore dynamics graphically is the so called phase space representation. Within this approach the dynamics of the variable(s) is shown as a vector field. In a one dimensional dynamics the phase space is a phase line, in a two dimensional case the phase space is a phase plane. Obviously the phase space representation only works for analysis of low dimensional dynamics because above three dimensions, the whole phase space cannot be depicted. Using the same dynamics as in Figure 10.1, we can construct a phase line (Fig. 10.2A), where $\dot{S} = 0$ in the origin, while $\dot{S} > 0$ and $\dot{S} < 0$ right and left of the origin. This shows that there is an unstable equilibrium at the origin, $S = 0$ because as indicated by the arrows/vectors, any small perturbation away from the origin will lead to movement towards right or left on the

line. This equilibrium can be compared to the unstable equilibrium of a marble sitting on the top of a hill; any small perturbation in the marble's position will cause it to roll downhill. If we redo the analysis for the case $\dot{S} = -S$, we get a fairly similar result (Fig. 10.2B). However in this case the equilibrium at the origin is stable; any perturbation away from the origin results in a movement back to the origin. This is similar to a marble located at the bottom of a bowl where every displacement of the marble results in a movement back to the bottom of the bowl. It is common practice to indicate unstable fixed points with an open circle and stable ones with a filled circle (Fig. 10.2A, B).

Fig 10.2

Now let us move on to a two dimensional case and use Equation (10.8) $\ddot{S} + b\dot{S} + cS = 0$ again. If we simplify this further by setting $b = 0$ and $c = 1$. The physicist immediately recognizes the expression as one without damping term, i.e. $b = 0$. We can decompose the 2nd order ODE into a pair of really simple 1st order ODEs (see also Section 9.5 for a MATLAB simulation in the phase plane):

$$\begin{aligned} \dot{S_1} &= \quad S_2 \\ \dot{S_2} &= -S_1 \,. \end{aligned} \qquad (10.1)$$

Now we can depict the flow in the phase plane of $S_1,\ S_2$. The S_1 axis (where $S_2 = 0$) is precisely the border where $\dot{S_1} = 0$ and where $\dot{S_1} > 0$ and $\dot{S_1} < 0$ above and below. The line where $\dot{S_1} = 0$ is also known as the $\boldsymbol{S_1}$ **nullcline**; here it is a coincidence that the S_1 nullcline is also the S_1 axis. Similarly, in this example, the S_2 axis is also the $\boldsymbol{S_2}$ **nullcline**. Only in this case, due to the minus sign, the direction of the flow of S_2 is positive for negative S_1 and vice versa. Combining the flow for $S_1,\ S_2$ (red and green arrows, Fig. 10.2C), we obtain a clockwise movement depicted by the blue circle in Figure 10.2C.

10. 2 Numerical Solution of an ODE

In the previous Chapter we showed that in some cases an ODE can be solved analytically. Sometimes, this can be challenging or even impossible. In these cases, numerical solution of the dynamics is the solution. The most common approach for obtaining numerical solutions describing the dynamics of a system is to divide time into a **fixed time step**. The evolution of the system is then computed over this time step, allowing one to determine the state of the system at the end of the fixed time step. Then the numerical procedure is repeated for a next time step. The idea underlying this procedure can be visualized by the solution drawn in Figure 10.1: one starts at some known condition (* in Fig. 10.1) and follows the direction of the field (indicated by the lines) for a short time step, after reaching the end point of that time step, the procedure is repeated. In this case one might evolve the system in two directions: the past (green line, Fig. 10.1) and the future (red line, Fig. 10.1). Obviously, any numerical method will be associated with some error: one only approximates the dynamics when following the straight line over the time step, and (as in any numerical procedure) there will be rounding errors. The first type of error depends on the size of the time step and in fast evolving systems it may have a tendency to grow with time.

The simples fixed time step method is the so called Euler's method. Let's go back to the example in Fig. 10.1 to demonstrate the principle: i.e. $\dot{S} = S$. Further, let's assume just as in Figure 10.1 that the initial condition at $t = 0$, indicated by S_0 is 1. Using this initial value and using a fixed time step $\Delta t = 0.5$s, we can compute that S after one time step denoted by S_1 is:

$$S_1 = S_0 + \dot{S}_0 \times \Delta t = 1 + 1 \times 0.5 = 1.5$$

We can repeat this procedure for the next time steps and we will get the following numerical outcomes for time steps 0 to 4 (i.e. $t = 0.0, 0.5, 1.0, 1.5, 2.0$):

$$S_0 = 1.0000$$
$$S_1 = 1.5000$$
$$S_2 = 2.2500$$
$$S_3 = 3.3750$$
$$S_4 = 5.0625$$

If we compute the values for $t = 0.0, 0.5, 1.0, 1.5, 2.0$s directly from the analytical solution of the ODE, $S = e^t$, we would have gotten:

1.0000 1.6487 2.7183 4.4817 7.3891

Thus indeed the Euler's method generates an approximation that indeed becomes worse with time. In this example, our numerical approximation is systematically lower than the correct outcome, because we underestimate the slope of the graph governing the change across the time step, leading to an estimate below the correct value. Even worse, since we underestimate the outcome at the end of the time step, our mistake grows with each time step. One important improvement can be obtained by reduction of the time step, say using 0.01s instead of 0.1s (homework).

Another alternative to improve our prediction of the numerical approximation is to employ the improved Euler's method. Here we compute two slopes instead of one, and use the average of the two slopes to evolve the system over the time step. Applying this to the previous example we would compute for the first time step the slope at the start and at the end of the time step:

One of the most commonly used numerical procedures is the Runge-Kutta method. It employs a four step approach to evaluate the dynamics over a fixed time step.

The following MATLAB script, pr10_2.m summarizes the three different numerical procedures for the $\dot{S} = S$ dynamics.

```
% pr10_2.m
% Numerical Solution
% dS/dt=S

clear;
close all

dt=.5;
N=10;
t=0:dt:dt*N;
```

```
yA=exp(t);          % Analytical Solution, see Eq (9.9)
S(1)=1;

% Euler's Method
for n=1:N
   s1=S(n);
   S(n+1)=S(n)+dt*s1;
end;
yE=S;               % Euler Solution

% Improved Euler's Method aka second-order Runge-Kutta
% employs two estimates of the slope s1 and s2
for n=1:N
   s1=S(n);
   s2=S(n)++dt*S(n);
   S(n+1)=S(n)+(dt/2)*(s1+s2);
end;
yiE=S;              % Improved Euler Solution

% Fourth-order Runge Kutta
% employs four estimates of the slope s1-s4
for n=1:N
      s1=S(n);
      s2=S(n)+(dt/2)*s1;
      s3=S(n)+(dt/2)*s2;
      s4=S(n)+dt*s3;
      S(n+1)=S(n)+(dt/6)*(s1+2*s2+2*s3+s4);
end;
yRK=S;              % Fourth-order Runge Kutta Solution

% Euler's Method @ 5 x smaller steps
dt=.5/5;
N=10*5;
ts=0:dt:dt*N;
for n=1:N
   s1=S(n);
   S(n+1)=S(n)+dt*s1;
end;
yEs=S;              % Euler Solution with smaller time step


% Plot Results
figure;hold;
plot(t,yA,'ko-')
plot(t,yE,'b.-')
plot(t,yiE,'r.-')
```

```
plot(t,yRK,'g.-')
plot(ts,yEs,'b*')

xlabel('time')
ylabel('S = exp (t)')
title('Black o-Analytical Outcome; Blue-Euler (* small steps); Red-Improved Euler; Green-4th Order RK')

% Summarize and print the findings
Y=[yA;yE;yiE;yRK;yEs(1:5:N+1)]'
```

When again comparing the first 4 time steps across the different methods we can see the difference in accuracy (Table 10.1).
Homework: Run the MATLAB script with different time steps.

Table 10.1

Without any special adaptation, numerical methods can applied to **nonlinear** ODEs that are difficult or impossible to solve. Let us look again at the example of a growth/birth process $\dot{S} = rS$, characterized by rate constant r. Now we add limiting resources and death to the dynamics of the population development. The result is that there is a target size, also known as **carrying capacity** K for the population. Below the target size, the population grows and above it the size decreases. This is described by the well-known nonlinear logistic equation, first constructed by Pierre-François Verhulst in the early 1800s (and rediscovered about a century later):

$$\dot{S} = rS(1 - \frac{S}{K}) \qquad (10.2)$$

For convenience in the remainder of our analysis we will use $r = 1$. It can be seen in Equation (10.2) that:

(1) $\dot{S} = 0$ if $S = K$,
(2) $\dot{S} > 0$ if $S < K$, and
(3) $\dot{S} < 0$ if $S > K$.

Thus at any size not equal to K, the system will change size to attain target size K.

The logistic equation is an example of a nonlinear ODE that can be solved analytically. We start by rewriting Equation (10.2) as (note that we used $r = 1$):

$$\frac{KdS}{S(K-S)} = dt,$$

And we use partial fraction expansion to get:

$$\left(\frac{1}{S} + \frac{1}{K-S}\right) dS = dt.$$

Now we integrate the terms followed by some algebra:

$$\log|S| + \log|K - S| = t + C,$$
$$\log\left|\frac{S}{K-S}\right| = t + C,$$
$$\left|\frac{S}{K-S}\right| = e^{t+C} = e^t e^C.$$

Here e^C is a constant. If we now use an initial condition S_0 at $t = 0$, we find a value $\frac{S_0}{K-S_0}$ for the constant. Here and in the following we drop the absolute values since we allow the constant to be positive as well as negative. If we now solve for S and do a bit of algebra:

$\frac{S}{K-S} = \frac{S_0}{K-S_0} e^t$, and we find:

$$S = \frac{KS_0}{S_0 + (K-S_0)e^{-t}} \ . \qquad (10.3)$$

In Equation (10.3) we can see that for $t \to \infty$ the size S becomes equal to the carrying capacity K. It is interesting that we didn't have to go through all this algebra to find this result because we could directly see in the ODE (Equation (10.2)) that the ultimate equilibrium occurs at $S = K$. Another intuitive way to look at the dynamics governed by the logistic equation is to plot the directions in the same way we did in Figure 10.1. The example depicted in Figure 10.3 is computed for a carrying capacity of six. Using the same initial condition as in Fig. 10.1, we now see that the growth is not unlimited but stabilizes at $K = 6$.

Fig 10.3

10.3 Partial Differential Equations

Partial differential equations (PDEs) are used to describe the dynamics of a metric with respect to different variables. An obvious example is a description of spatiotemporal dynamics. For instance a propagating brain wave is a potential field that changes with both time and location. The essence is that a PDE includes separate derivatives to describe dependence to time and location. Ordinary derivatives consider metrics that only depend on a single variable, e.g. with respect to time t or location x, reflected by notations dS/dt or dS/dx. In contrast when metric S depends on both time and location, the PDE employs partial derivatives. An example of a PDE is

$$\partial^2 S/\partial x^2 = K\ \partial S/\partial t. \qquad (10.4)$$

This type of equation with S being a function of place x and time t plays a role in processes such as diffusion or conduction of heat. Sometimes these PDEs can be solved by transforming them into ODEs, e.g if we assume that the dependence of S to location and time can be written as:

$$S(x,t) = u(x)v(t). \qquad (10.\ 5)$$

If this is possible, Equation (10.4) can be replaced by two ODEs by using the following procedure. Using Equation (10.5), we can define

$$\partial S/\partial t = u\, dv/dt,\ \partial S/\partial x = v\, du/dt, \text{ and } \partial^2 S/\partial x^2 = v\, d^2u/dx^2.$$

Plugging this into Equation (10.4) gives:

$$v\, d^2u/dx^2 = K\, u\, dv/dt,$$
$$\frac{1}{u} d^2u/dx^2 = \frac{K}{v}\, dv/dt.$$

Note that in the last expression the part left of the equal sign is uniquely a function of location x, and the part right of the equal sign only depends on time t. Since they both have to be equal, say to a value k we obtain two ODEs:

$$\frac{1}{u} d^2u/dx^2 = k \quad \text{or} \quad d^2u/dx^2 - ku = 0, \text{ and}$$
$$\frac{K}{v}\, dv/dt = k \quad \text{or} \quad dv/dt - \frac{k}{K} v = 0.$$

Following our assumption in Equation (10.5), the product of the solutions to each of these ODEs is a solution to the PDE.

Just as solving ODEs numerically, finding numerical solutions by simulation of the PDE is a good approach to examine the dynamics. An example is shown in ***CH 31*** where we show spatiotemporal activity of a cortical area (pr31_TBD and Fig. 31.TBD). To obtain **numerical stability** in simulations of a PDE, it is important that the step sizes employed for both the temporal and spatial domains satisfy the so-called **von Neumann criterion** (e.g. Press et al., 1992). This criterion describes the required relationship between the spatial and time steps in the numerical solution:

$$\frac{2\Delta t}{K\Delta x^2} \leq 1 \qquad (10.6)$$

Here, Δx and Δt are the spatial and time steps used for the numerical solution, and K is the constant from Equation (10.4). If you forget about this and violate this criterion, you will be the first to know since your solutions will blow up.

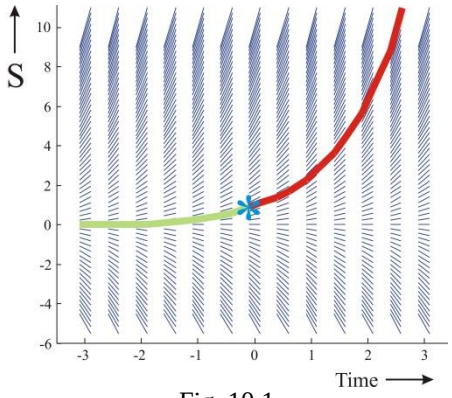

Fig. 10.1

Dynamics of the ODE $\dot{S} = S$.The flow of S plotted against the time axis. The initial condition is depicted by the blue * and the (hand-drawn) associated solution for negative and positive times by the green and red lines. The flow lines in this Figure were obtained with MATLAB script pr10_1.m

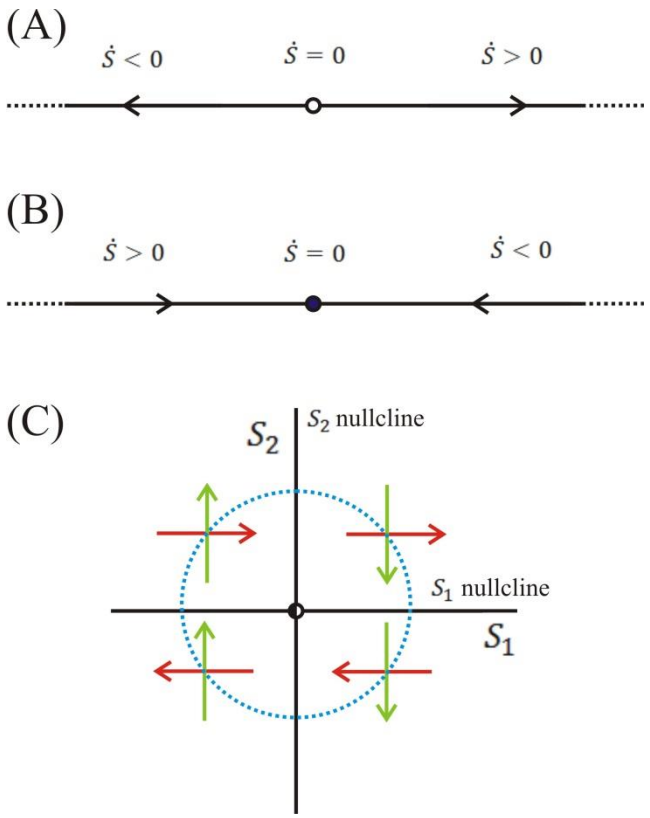

Fig 10.2

Phase space plots for a one and two dimensional case. The phase lines in panels (A) and (B) show the flow of $\dot{S} = S$ and $\dot{S} = -S$ respectively. The phase plane in panel (C) shows the flow for $\ddot{S} + S = 0$. Further explanation in the text.

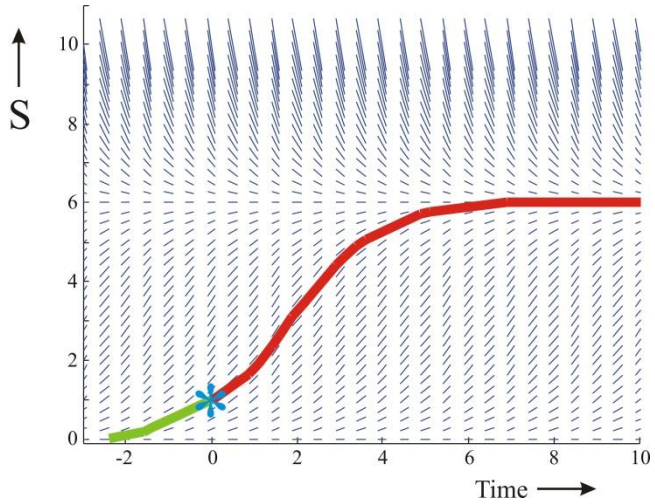

Fig. 10.3

The logistic equation $\dot{S} = S(1 - \frac{S}{6})$. The flow of S plotted against the time axis. The initial condition is depicted by the blue * and the (hand-drawn) associated solution for negative and positive times by the green and red lines. The flow lines in this Figure were obtained with MATLAB script pr10_3.m

Table 10.1 Comparison of numerical solvers for $\dot{S} = S$

Time(s)	exp(t)	Euler's	Improved Euler's	4^{th} Order Runge-Kutta
0.0	1.0000	1.0000	1.0000	1.0000
0.5	1.6487	1.5000	1.6250	1.6484
1.0	2.7183	2.2500	2.6406	2.7173
1.5	4.4817	3.3750	4.2910	4.4794
2.0	7.3891	5.0625	6.9729	7.3840