Numerical Solutions and Direction Fields for Systems

Kurt Bryan and SIMIODE

A script to illustrate how to solve systems of ODEs numerically, and to draw a direction field for a pair of autonomous ODEs in Maple.

Example: We have a pair of ODEs of the form (for example) x1' = x1 - x2^2 and x2' = x1*x2+x1, where x1 and x2 are functions of t. The system can be written in a vector-valued form as d**x**/dt = **f**(t,**x**) where

```
f = @(t,x) [x(1)-x(2).^2; x(1).*x(2)+x(1)];
```

The function f accepts a scalar argument "t" and vector argument "x", and returns a column vector with the same dimension as x.

Numerical Solution: We can solve the system numerically with data x1(0) = 4, x2(0) = 1 on the interval 0 <= t <= 2, with the command

```
[tvals, xvals] = ode45(f, [0 2], [4 1]);
```

(this uses Matlab's general purpose "ode45()" numerical solver---many others are available.) The solution components can be plotted as

```
plot(tvals,xvals(:,1),'-r',tvals,xvals(:,2),'-b')
```

A parametric plot can be constructed with

```
plot(xvals(:,1)',xvals(:,2)')
```

Direction Fields: If the ODEs are autonomous (d**x**/dt = f(**x**), no dependence of **f** on t) we can sketch a direction field for some range of x1 and x2. It's probably easiest to first define the components of **f**(**x**) separately first, as

```
f1 = @(x1,x2) x1 - x2.^2; %Don't forget to specify operations element-wise with a period.
f2 = @(x1,x2) x1.*x2 + x1;
```

where we can omit "t" since we assume the system is autonomous.

Then to sketch a direction field on the range -5 < x1, x2 < 5, execute

```
[X1,X2] = meshgrid(-5:0.5:5, -5:0.5:5);
```

which sets up a grid in the (x1,x2) plane on the given region with 0.5 spacing between grid points in each direciton. Then compute the direction vector "v" at each grid point with

```
v1 = f1(X1,X2);
v2 = f2(X1,X2);
len = sqrt(v1.^2 + v2.^2); %Rescale arrows to all have same length---typically looks better
v1 = v1./len;
v2 = v2./len;
```

Have then Matlab plot arrows at the appropriate grid points

```
quiver(X1,X2,v1,v2,0.5); %0.5 defines arrow lengths---adjust to taste.
axis([-5 5 -5 5])
```

We can solve for the fixed points and have Matlab plot a solid dot at each (if the ODE right hand sides are simple enough):

```
syms p1 p2;
sol = solve([f1(p1,p2)==0,f2(p1,p2)==0],[p1,p2]);
n = length(sol.p1);
hold on
scatter(sol.p1,sol.p2,'filled','MarkerFaceColor',[0 0 0]);
drawnow
```

A solution(s) with given initial data can be superimposed by solving numerically and overlaying the solution. For example, if x1(0) = 4, x2(0) = 1

```
[tvals, xvals] = ode45(f, [0 2], [4 1]); %Solve forward in time from t = 0
plot(xvals(:,1)',xvals(:,2)','-r')
[tvals, xvals2] = ode45(f, [0 -2], [4 1]); %Solve backward in time from t = 0
plot(xvals2(:,1)',xvals2(:,2)','-r')
hold off
```

A Phase Portrait Command: The supplied command "draw_auto_phase" accepts two functions f1(x1,x2) and f2(x1,x2) as above that define an autonomous ODE system in the plane, along with a third argument "viewrange" of the form [x1low x1 high x2low x2high] that defines the region for the phase portrait. For the above system the phase portrait can be drawn with

```
draw_auto_phase(f1,f2,[-5 5 -5 5])
```

This draws the direction field as above, and the fixed points (if they can be found algebraically). We can also add solutions through any number of initial points by supplying an additional n x 2 matrix, where each row of the matrix is the (x1,x2) initial point. For example,

```
draw_auto_phase(f1,f2,[-5 5 -5 5],[0 1; 0 2; 4 -2])
```

The solutions trajectories are sketched both forward and backward in time.