Part IV. Linear Differential Equations

Section 3. Two Dimensional Systems

A second order differential equation in normal form

$$y'' = f(t, y, y')$$

can always be converted into an equivalent system of two first order equations

$$y' = v$$

$$v' = f(t, y, v)$$

If the function *f* does not depend upon *t*, then the second order equation (and the system) is called autonomous. More generally, a two dimensional system is a pair of first order differential equations of the form

$$x' = f (x, y, t)$$

$$y' = g (x, y, t)$$

The primes denote differentiation with respect to *t*. If neither *f* nor *g* depend upon *t* the system is autonomous. In this section *Mathematica* is used to analyze the solutions to such systems. By now you are familiar with the tools we need: DSolve, NDSolve, PlotVectorField, and ParametricPlot.

```
<<Graphics`PlotField`
```

The solution to a two dimensional system is a pair of functions $x(t)$ and $y(t)$. Solution behavior is analyzed using time series and plots of phase plane trajectories of the form $(x(t), y(t))$ when the system is autonomous. State space trajectories $(x(t), y(t), t)$ can be used when the system is not. We consider the autonomous case first

$$x' = f (x, y)$$

$$y' = g (x, y)$$

See Ledder, Chapter 5, Section 3.

Fields and flows

Either DSolve (or NSSolve) will output time series for a two dimensional system. Trajectories can be obtained using the phi function (or Table) with ParametricPlot. If the system is autonomous PlotVectorField will also draw a direction field. Consider the following example derived from the Lotka-Volterra predator prey model (Ledder, page 306, Example 2).

```
DEsystem = {x'[t] == x[t]*(1 - y[t]),
            y'[t] == y[t]*(x[t] - 1)}
{x′[t] == x[t] (1 - y[t]), y′[t] == (-1 + x[t]) y[t]}
```

This is a non-linear system and exact solution formulas are very hard to obtain. *Mathematica*'s DSolve function is unable to obtain solutions for the initial conditions $x(0) = 0.4$ and $y(0) = 0.4$..

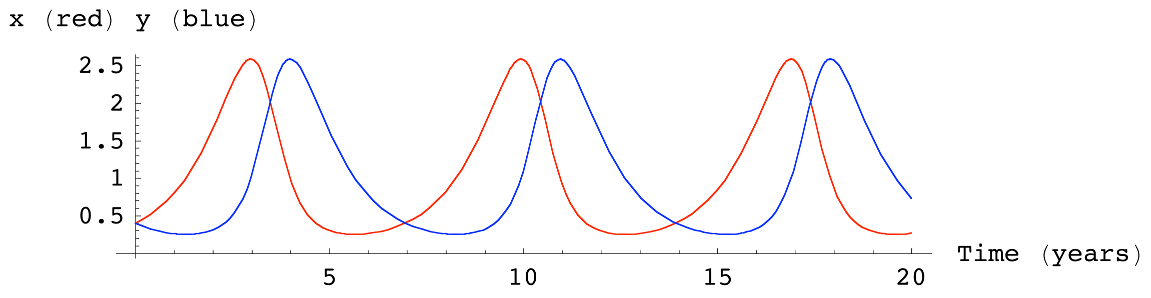
```
DSolve[ {DEsystem, x[0]==0.4, y[0]==0.4}, {x, y}, t ]

 InverseFunction::ifun : Inverse functions are being
    used. Values may be lost for multivalued inverses. More…

 Solve::ifun :
  Inverse functions are being used by Solve, so some solutions may
    not be found; use Reduce for complete solution information. More…

 Solve::tdep : The equations appear to involve the variables
    to be solved for in an essentially non-algebraic way. More…

 Solve::tdep : The equations appear to involve the variables
    to be solved for in an essentially non-algebraic way. More…
$Aborted
```

We turn to NDSolve, requesting a numerical solution for the interval from $t = 0$ to $t = 20$..

```
soln = NDSolve[ {DEsystem,x[0]==0.4,y[0]==0.4}, {x, y}, {t,0,20} ]
{{x → InterpolatingFunction[{{0., 20.}}, <>],
  y → InterpolatingFunction[{{0., 20.}}, <>]}}
```

The following plot displays the time series, $x(t)$ and $y(t)$. We get fancy with informative axes labels.

```
Plot[ Evaluate[{x[t],y[t]}/.soln], {t,0,20}, AspectRatio->1/4,
      PlotStyle->{{RGBColor[1,0,0]},{RGBColor[0,0,1]}},
      AxesLabel->{"Time (years)","x (red) y (blue)"} ]
```

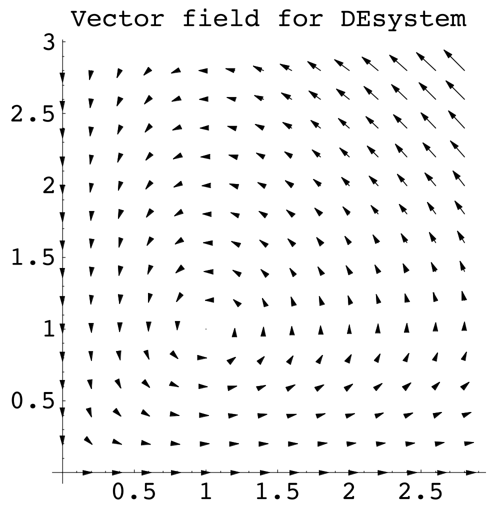


The system is autonomous so PlotVectorField can be used to display the vector field:

At point (x , y) plot the vector $< (x (1 - y) , y (x - 1) >$

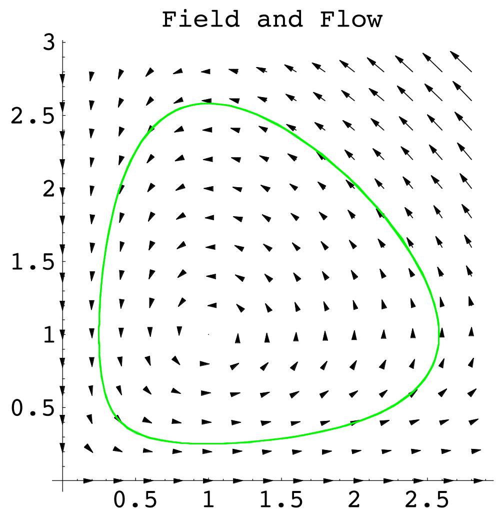
The plot is named "vf" so it can be displayed with the trajectory below..

```
vf = PlotVectorField[ {x*(1-y),y*(x-1)}, {x,0,2.8}, {y,0,2.8}, Axes->True,
PlotLabel->"Vector field for DEsystem" ]
```



And here is the trajectory, making its way through the vector field. The solution is periodic.

```
Show[ vf,
     ParametricPlot[ Evaluate[{x[t],y[t]}/.soln], {t,0,20},
                    PlotStyle->RGBColor[0,1,0] ],
     PlotLabel->"Field and Flow" ]
```



Estimating the period: Numeric data

We can estimate the period using a table of numeric data generated from the NDSolve solution. It appears from the time series plot above that the solutions start to repeat at about $t = 7$. The following Table pins it down to 6.96.

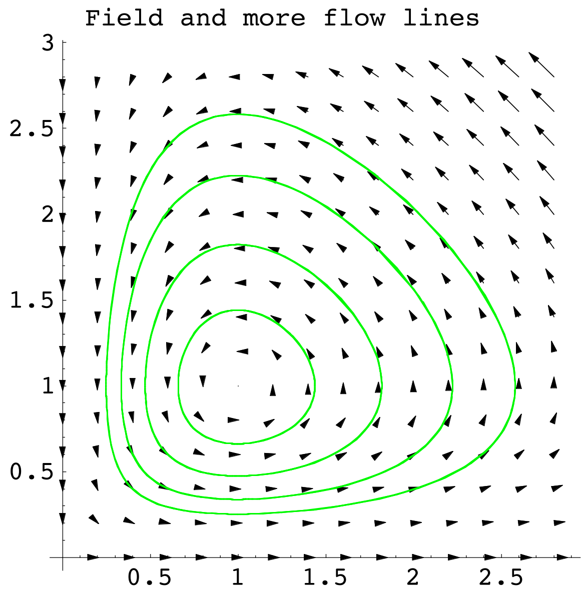
```
Join[ {{t,x[t],y[t]}},
      Evaluate[ Table[ {t,x[t],y[t]}, {t,6.9,7,0.01}]]/.soln[[1]] ]
      //MatrixForm
```

```
⎛  t      x[t]      y[t]   ⎞
⎜ 6.9   0.385623  0.415293 ⎟
⎜ 6.91  0.387889  0.412754 ⎟
⎜ 6.92  0.390179   0.41024 ⎟
⎜ 6.93  0.392492  0.407751 ⎟
⎜ 6.94  0.394828  0.405286 ⎟
⎜ 6.95  0.397188  0.402845 ⎟
⎜ 6.96  0.399572  0.400429 ⎟
⎜ 6.97  0.401979  0.398037 ⎟
⎜ 6.98  0.404411  0.395668 ⎟
⎜ 6.99  0.406867  0.393324 ⎟
⎝  7.   0.409348  0.391002 ⎠
```

A more complete flow

A more complete picture of the flow is obtained with a family of numeric solutions. The following input defines four initial values, creates four numeric solutions,and then shows their parametric plots in the vector field. The picture is named flow so it can be shown again below.

```
inits = {0.4, 0.6, 1.3, 1.8};
solns = Evaluate[
        Table[ NDSolve[ {DEsystem,x[0]==inits[[k]],y[0]==inits[[k]]},
               {x,y}, {t,0,20} ], {k,4} ] ];
Show[ vf,
      ParametricPlot[ Evaluate[Table[{x[t],y[t]}/.solns[[k]], {k,4}]],
                      {t,0,20}, PlotStyle->RGBColor[0,1,0] ],
                      PlotLabel->"Field and more flow lines",
                      AspectRatio->1/1 ]
flow = %;
```



Nullclines, critical points

The nullclines for an autonomous system are the curves determined by the equations

$$f(x, y) = 0$$

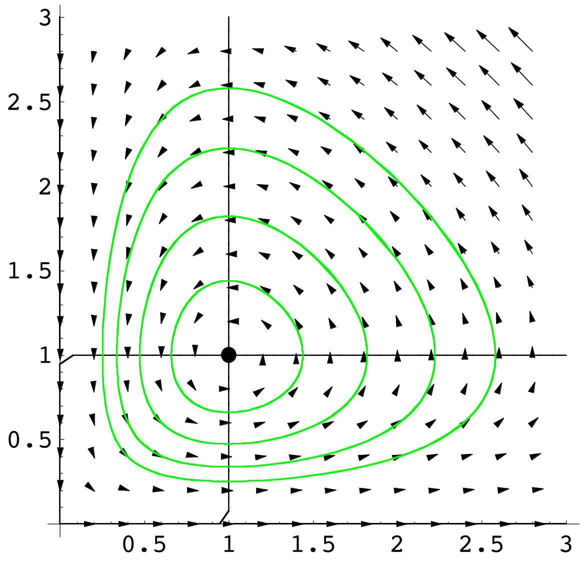
$$g(x, y) = 0$$

The points where the nullcllines cross are called the critical points or the stationary points of the system.

The ImplicitPlot function is used to draw the nullclines for DEsystem. The critical point in the first quadrant is also drawn using the ListPlot function. These plots are given names (output is suppressed) and shown with the above flow plot. We first load the ImplicitPlot function

```
<< Graphics`ImplicitPlot`
```

```
nullclines = ImplicitPlot[ {x*(1-y)==0, y*(x-1)==0}, {x,0,3}, {y,0,3} ];
stationary = ListPlot[ {{1,1}}, PlotStyle→PointSize[0.03] ];
Show[ nullclines, stationary, flow, AspectRatio->1/1 ]
```



Symbolic solutions, integral curves

The Lotka-Volterra system is non-linear. *Mathematica* cannot obtain a symbolic solution for the trajectories. However, formulas for curves determined by the trajectories can be obtained by eliminating the *t* variable to make a first order differential equation for the *y* variable as a function of *x*. Solutions to this differential equation are called "integrals" of the system, their graphs are called integral curves. Integral curves for Lotka-Voltera can easily be found "by hand" (Ledder, page 306). *Mathematica*'s computations are shown in the next few entries.

Create DE:

```
DE = y'[x] == DEsystem[[2,2]]/DEsystem[[1,2]]
```

$$y'[x] == \frac{(-1 + x[t])\, y[t]}{x[t]\, (1 - y[t])}$$

Change y[t] and x[t] into y[x] and x respectively.

```
DE = DE/.{x[t]->x, y[t]->y[x]}
```

$$y'[x] == \frac{(-1 + x)\, y[x]}{x\, (1 - y[x])}$$

Solve DE,with the initial condition y(0,4) = 0.4. to see if DSolve is up to the job.

```
soln = DSolve[ {DE, y[0.4]==0.4}, y, x ]
```

```
InverseFunction::ifun : Inverse functions are being
   used. Values may be lost for multivalued inverses. More…

Solve::ifun :
 Inverse functions are being used by Solve, so some solutions may
   not be found; use Reduce for complete solution information. More…

Solve::ifun :
 Inverse functions are being used by Solve, so some solutions may
   not be found; use Reduce for complete solution information. More…
```

$$\{\{y \to \text{Function}[\{x\}, -\text{ProductLog}[-\frac{e^{-2.63258+x}}{x}]]\}\}$$

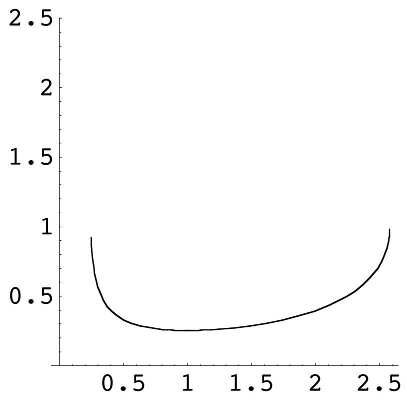
Mathematica has found an explicit solution in terms of the "ProductLog" function.

```
?ProductLog
```

```
ProductLog[z] gives the principal solution for w in
   z = w e^w. ProductLog[k, z] gives the kth solution. More…
```

The plot of the solution shows that DSolve has obtained the formula for the bottom half of the integral curve for the system.

```
Plot[ y[x]/.soln, {x,0,3}, PlotRange->{0,2.5}, AspectRatio->1/1 ]
```



Linear systems, linear flows

Two dimensional linear systems have the form

$$x' = a\,x + b\,y + f$$

$$y' = c\,x + d\,y + g$$

where *x, y, a, b, c, d, f,* and *g* are all functions of *t*. The functions *f* and *g* are sometimes referred to as the forcing functions. If they are both zero, the system is called homogeneous (or unforced).

When *a, b, c*, and *d* are constants, symbolic solutions can be always be found, provided the functions *f* and *g* are elementary (products of polynomials, sine, cosine, exp, etc.). Such systems are important not only because they arise in many applications, but also because they are used to approximate non-linear systems in regions near their critical points.

Example Consider the autonomous linear system

$$x' = x - 2\,y + 3$$

$$y' = x + y - 2$$

Sketch the flow, nullclines and stationary point. Find symbolic solutions.

```
DEsys = { x'[t] == x[t] - 2*y[t] + 3,
          y'[t] == x[t] +   y[t] - 2 }
{x′[t] == 3 + x[t] - 2 y[t], y′[t] == -2 + x[t] + y[t]}
```

Let's find the stationary point first. The Solve function can be used. Note that the equations and the unknowns must be entered in separate lists. The output is a set, or sets, of arrow substitutions..

```
stationary = Solve[ {x-2*y+3==0, x+y-2==0}, {x,y} ]
```

$$\left\{\left\{x \to \frac{1}{3},\ y \to \frac{5}{3}\right\}\right\}$$

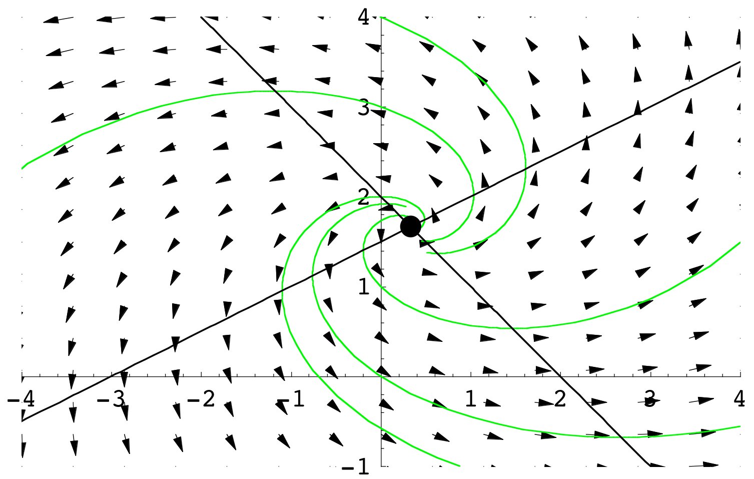
Use the solution to make a "point" (i.e, a list for plotting, as follows.

```
statpoint = {x,y}/.stationary
```

$\{\{\frac{1}{3}, \frac{5}{3}\}\}$

The next entries create the plots for vector field, the flowlines (using NDSolve), the nullclines, and the stationary point. They are all shown together using Show.

```
vf = PlotVectorField[{x-2*y+3, x+y-2}, {x,-4,4}, {y,-1,4} ];
solns = Table[ NDSolve[ {DEsys, x[0]==0, y[0]==k}, {x,y}, {t,-2,3} ],
               {k,0,4} ];
flow=ParametricPlot[Evaluate[Table[{x[t],y[t]}/.solns[[k,1]],{k,1,5}] ],
                   {t,-2,3}, PlotStyle->RGBColor[0,1,0] ];
nullclines = ImplicitPlot[ {x-2*y+3==0,x+y-2==0}, {x,-4,4}, {y,-1,4} ];
stationarypoint = ListPlot[ statpoint, PlotStyle->PointSize[0.03] ];
Show[ {vf, flow, nullclines, stationarypoint}, Axes->True,
       PlotRange->{{-4,4},{-1,4}}, AspectRatio->5/8 ]
```



This kind of outward spiraling autonomous flow is called a "source". Phase portraits for constant coefficient autonomous linear systems are often referred to as "linear flows".

Symbolic solution formulas for a source are given in terms of sines, cosines, and exponential functions.

```
solnFormulas = DSolve[ DEsys, {x,y}, t ]
```

$$\Big\{\Big\{x \to \text{Function}\Big[\{t\},\ e^{t}\,C[1]\,\text{Cos}\big[\sqrt{2}\ t\big] - \sqrt{2}\ e^{t}\,C[2]\,\text{Sin}\big[\sqrt{2}\ t\big] + \frac{\text{Sin}\big[\sqrt{2}\ t\big]\,\big(-10\,\text{Cos}\big[\sqrt{2}\ t\big] + \sqrt{2}\ \text{Sin}\big[\sqrt{2}\ t\big]\big)}{3\sqrt{2}} + \frac{1}{3}\,\text{Cos}\big[\sqrt{2}\ t\big]\,\big(\text{Cos}\big[\sqrt{2}\ t\big] + 5\sqrt{2}\ \text{Sin}\big[\sqrt{2}\ t\big]\big)\Big],$$

$$y \to \text{Function}\Big[\{t\},\ e^{t}\,C[2]\,\text{Cos}\big[\sqrt{2}\ t\big] + \frac{e^{t}\,C[1]\,\text{Sin}\big[\sqrt{2}\ t\big]}{\sqrt{2}} - \frac{1}{6}\,\text{Cos}\big[\sqrt{2}\ t\big]\,\big(-10\,\text{Cos}\big[\sqrt{2}\ t\big] + \sqrt{2}\ \text{Sin}\big[\sqrt{2}\ t\big]\big) + \frac{\text{Sin}\big[\sqrt{2}\ t\big]\,\big(\text{Cos}\big[\sqrt{2}\ t\big] + 5\sqrt{2}\ \text{Sin}\big[\sqrt{2}\ t\big]\big)}{3\sqrt{2}}\Big]\Big\}\Big\}$$

These formulas are easier to understand if displayed in simplified form.

```
x[t]/.solnFormulas//FullSimplify
```

$$\Big\{\frac{1}{3} + e^{t}\,\big(C[1]\,\text{Cos}\big[\sqrt{2}\ t\big] - \sqrt{2}\ C[2]\,\text{Sin}\big[\sqrt{2}\ t\big]\big)\Big\}$$

```
y[t]/.solnFormulas//FullSimplify
```

$$\Big\{\frac{5}{3} + e^{t}\,C[2]\,\text{Cos}\big[\sqrt{2}\ t\big] + \frac{e^{t}\,C[1]\,\text{Sin}\big[\sqrt{2}\ t\big]}{\sqrt{2}}\Big\}$$

Observe that the "stationary solution" is obtained by setting the constants equal to 0. Linear flows are discussed in more detail in the next section.