New Options to Visualize Systems of Differential Equations in Maple *

Mohammad ali Ebrahimi and Michael Monagan

Department of Mathematics, Simon Fraser University
mebrahim@cecm.sfu.ca and mmonagan@cecm.sfu.ca

Abstract

The DEplot command in Maple is used to show the direction field and solution curves of a differential equation or system of differential equations. We present new options created for the DEplot command for enhancing the visualization of a direction field and for animating a direction field and the solution curves in time. We present a new Maplet, DEplotlet(DEplotInteractive), which simplifies the use of the DEplot command. We have also designed a database of different systems of differential equations which can be easily modified by the instructor. The database includes the Lotka-Volterra predator-prey system and the Kermack-McKendric epidemic model. It includes default choices for the parameters and for initial values so that the instructor and student can obtain a good plot with a single mouse click.

1 Introduction:

Consider the autonomous system $x'(t) = f(x, y)$, $y'(t) = g(x, y)$ where $x, y \in \mathbb{R}$. One can draw small line segments with a slope $\frac{g(x_i,y_i)}{f(x_i,y_i)}$ at any desired point (x_i, y_i). Each line segment is then tangent to the solution at the point (x_i, y_i). The set of all these line segments is called the direction field. By plotting the direction field of the system one can get a good approximation of the solution and its properties. In addition plotting direction fields is fast and easy, since there is no need to solve ordinary differential equation('s). Direction fields can also be used for testing the accuracy of the numerical solutions. For example, the Lotka-Volterra system

$$x'(t) = \alpha\, x(t) - b\, x(t)y(t),\ y'(t) = -\beta\, y(t) + c\, x(t)y(t) \tag{1}$$

is a predator-prey model. Here, $x(t)$ is the population of the prey at time t, $y(t)$ is the population of the predator at time t, $\alpha > 0$ is the birth rate of the prey, $\beta > 0$ is the death rate of the predator and the terms $c\, x(t)y(t)$ and $-d\, x(t)y(t)$ where $c > 0$ and $d > 0$ describe the interaction of the predator and prey. Throughout the paper we use the following system for showing plots.

$$des := [x'(t) = x(t)(1 - y(t)),\ y'(t) = .3y(t)(-1 + x(t))].$$

Figure 1 shows a plot of the system obtained using Maple's DEplot command. Notice that instead of drawing line segments, Maple shows little arrows to indicate the direction of the solution at the midpoint of the arrow.

*Supported by NSERC of Canada and the MITACS os Canada.

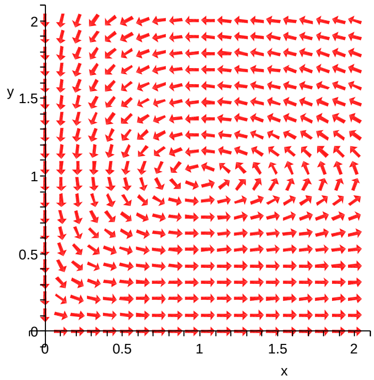


Figure 1: This is a direction field for the Lotka-Volterra model created by the Maple command `DEplot`. The Maple command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0..2, y=0..2, arrows=large);`

In this paper we show new options and tools created to enhance the visuals of the direction fields for the system of differential equations. In section 2 we discuss options to show the velocity of the solutions, in section 3 we give options to improve the visualization of the direction field, in section 4 we give two new arrow styles to improve the approximation of the direction field, and in section 5 options for animating the direction field and solution curves. In section 6 we describe the `DEplotInteractive` Maplet and the database of systems of differential equations.

2 Velocity Options

In Figure 1, we can see the "direction" but not the velocity of the direction field. The velocity option determines the norm of the speed of the solution at each point. For example if we have the system $\{y'(t) = f(x,y), x'(t) = g(x,y)\}$ then the norm of the speed at the point (x,y) can be represented by $\sqrt{f(x,y)^2 + g(x,y)^2}$. We have used this information and created two options for showing the norm of the speed of the solution of a system of differential equation at different points in the `DEplot` command. Our first solution to present the velocity is by color. The second solution is by the size of the arrows. In order to create a direction field which can demonstrate the norm of the speed at the position of each arrow, first we need to find the largest value of the speed among all the values of speed of the arrows. This is so we can scale the velocities, so they are between zero to 1. After finding the largest value of the velocities then at each point, where the arrow is located, the final value of velocity is represented by the value of $\frac{\sqrt{f(x,y)^2+g(x,y)^2}}{MAX(\sqrt{f(x,y)^2+g(x,y)^2})}$.

2.1 color=velocity[hicolor, lowcolor]

In this option the color of the arrows is determined by the velocity of the field. The low and high velocity colors are specified as the index of velocity. The option `color = velocity` can be used with the default of `[green, red]`. To find the RGB color value of each grid point, we use the following formula:

$$\delta = \frac{\sqrt{f(x,y)^2 + g(x,y)^2}}{MAX(\sqrt{f(x,y)^2 + g(x,y)^2})}; \tag{2}$$

$$R = lowcolor + \delta(Rhicolor - Rlowcolor); \tag{3}$$

$$G = lowcolor + \delta(Ghicolor - Glowcolor); \tag{4}$$

$$B = lowcolor + \delta(Bhicolor - Blowcolor); \tag{5}$$

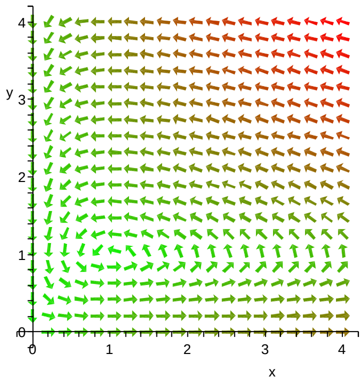


Figure 2: This is a direction field for the Lotka-Volterra model with the use of the `color = velocity` option. The Maple command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0..2, y=0..2, arrows=large, color=velocity);`

2.2 size=velocity

In `size = velocity`, the size of the arrows at each points indicates the velocity of the solution of that point. Smaller arrows indicate the smaller norm of speed and the larger arrows indicate the large norm of speed. In order to create different sizes of arrows we use the value of $\frac{\sqrt{f(x,y)^2+g(x,y)^2}}{MAX(\sqrt{f(x,y)^2+g(x,y)^2})}$ to find the length and width of each arrow. Velocities created this way appear poor because the small velocities show as tiny arrows, dots, so we scale them using a square root scaling. Even with this scaling, small velocities appear quite small. See Figure 3. In our opinion, the color option is better than the size option to demonstrate the velocity dimension of a direction field.

3 Direction Field Options

The option `dirfield` determines the placement of arrows for the direction field. By default Maple creates the direction fields with all the arrows on an equally spaced grid. The default number of points are 20 on the horizontal and 20 on the vertical axis, i.e. a 20 by 20 grid. One can change these values by the option `dirgrid=[` p`::posint,` q`::posint]` where p, the first element of `dirgrid`, represents the number of points on the horizontal axis and q, the second element of `dirgrid`, represents the number of points on the vertical axis. We have implemented two new options(aside from `dirgrid` option in the `DEplot` command), which the user can use in the `DEplot` command to determine the position of arrows. One option is to have the arrows of the direction field in random

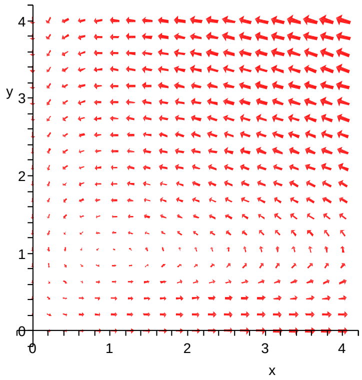


Figure 3: This is a direction field for the Lotka-Volterra model with the use of the `size = velocity` command. The Maple command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0..2, y=0..2, arrows=large, size=velocity);`

positions. The second option is that the user can determine the position of the arrows by a list of points.

3.1 dirfield=positive integer:

To enhance the visualization of the direction field one can use the new command `dirfield = n::posint` to create a direction field with n randomly positioned arrows. See Figure 4. In our opinion, random position direction field gives the best visual.

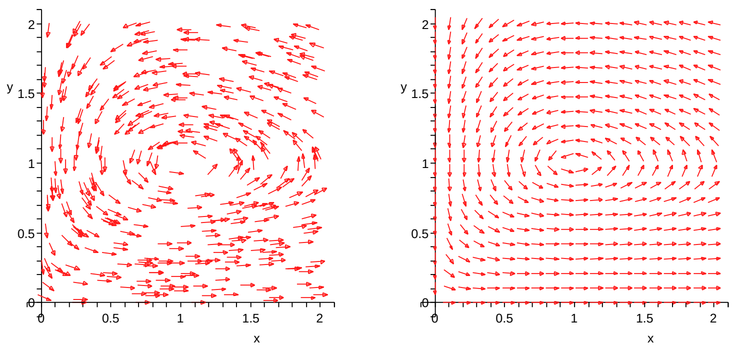


Figure 4: Direction field for the Lotka-Volterra model with the use of the `dirfield = 400` command. The Maple command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0..2, y=0..2, arrows=large, dirfield=400);` The second plot is the plot of the same system but in a rectangular grid. In comparison of the two plots we see that the random positioned field gives a better visual.

3.2 dirfield = [points]

In addition to the `dirfield = n::posint` option, we have implemented the the option `dirfield = [points]`. With this option one can assign the desired positions of the arrows manually. See Figure

5.

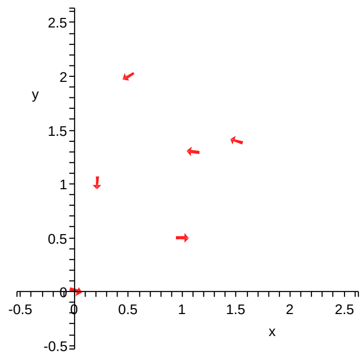


Figure 5: Direction field for the Lotka-Volterra model with the use of the `dirfield = [points]` command. The command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0..2, y=0..2, arrows=large, dirfield=[[1.1,1.3],[1.5,1.4],[0.21,1.01],[0.01,0.01],[1,0.5],[0.5,2]]);`

4 New Arrows:

To improve the visualization for the direction field, further, we have created new objects (fish shape) for which the curvature of each fish approximates the solution curve. The curvature of each fish is approximated by the modified Euler's method. These objects help the user to have a better approximation of the solution to the system of differential equations. See Figure 6.

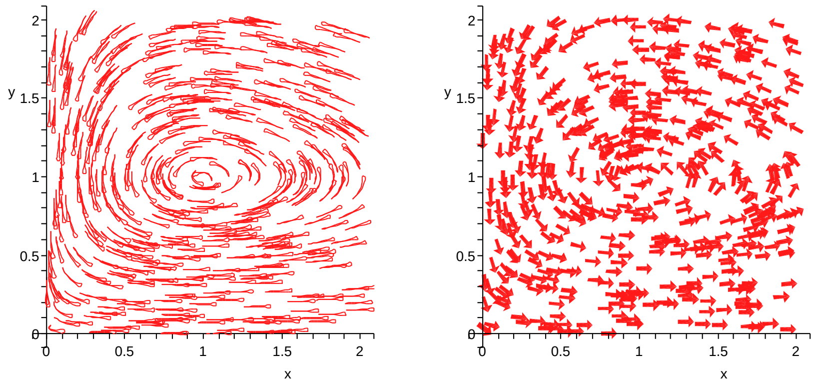


Figure 6: This is a direction field for the Lotka-Volterra model with the use of the `arrow = fish` command. The command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0..2, y=0..2, arrows=fish, dirfield=500);` In comparison of the two plots we see that the plot with fishes gives a sharper approximation of the solution.

4.1 Method:

Lets assume we want to create a fish at the point (a, b) for autonomous system $\{y'(t) = f(x, y), x'(t) = g(x, y)\}$. Evaluating the two functions $f(x, y)$ and $g(x, y)$ at the point (a, b) we set $\mu = \frac{f(a,b)}{g(a,b)}$, $(a', b') = (a, b) + \mu(dx, dy)$, and $(a'', b'') = (a, b) + -\mu(dx, dy)$ where dx and dy are small. The two points (a', b') and (a'', b'') are approximations of two points of the solution curve which passes through the point (a, b). One way to improve this approximation is to use the Euler's improved method. Hence, we do another function evaluation at (a', b') and at (a'', b'') and find $\omega = \frac{f(a',b')}{g(a',b')}$ and $\nu = \frac{f(a'',b'')}{g(a'',b'')}$ which leads us to find $(a_1, b_1) = (a, b) + (\frac{-\mu+\omega}{2})(dx, dy)$ and $(a_2, b_2) = (a, b) + (\frac{\mu+\nu}{2})(dx, dy)$. see Figure 7.

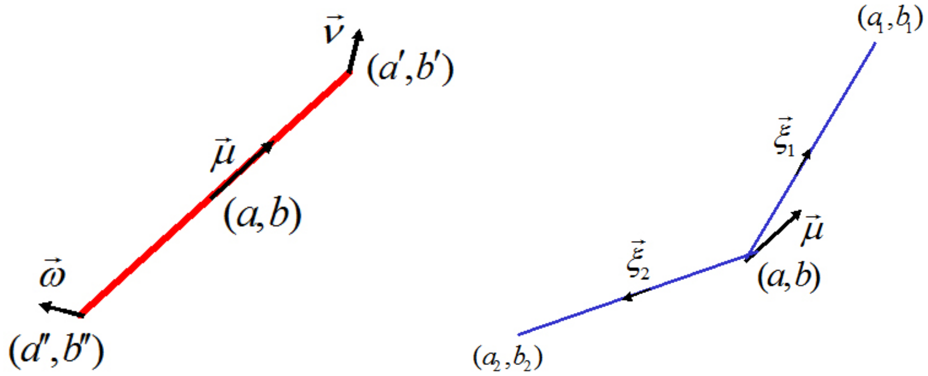


Figure 7: Steps to find the two end points $(a_1, b_1), and (a_2, b_2)$ of the fish via modified Euler's method. $\xi_1 = \frac{\mu+\nu}{2}$, $\xi_2 = \frac{-\mu+\omega}{2}$.

We now have three points (a, b), (a_1, b_1), and (a_2, b_2). To approximate the solution of $f(x, y), g(x, y)$ near $x = a, y = b$, we can find a quadratic curve which fits these three points. Hence, we have a parametric quadratic approximation with respect to $t \in [0, 2]$ of the solution curve passing through the point (a, b) where $x(0) = a$ and $y(0) = b$.

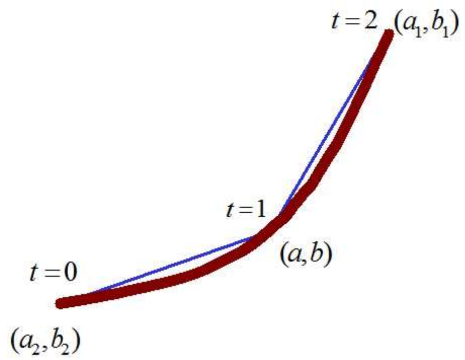


Figure 8: Parametric quadratic polynomial fit on the three points

The parametric solution to the quadratic curve which passes through the three points (a, b), (a_1, b_1), and (a_2, b_2) is given by:

$$x(t) = ((\frac{a_2}{2} - a + \frac{a_1}{2})t - \frac{3a_2}{2} + 2a - \frac{a_1}{2})t + a_2 \quad (6)$$

$$y(t) = ((\frac{b_2}{2} - b + \frac{b_1}{2})t - \frac{3b_2}{2} + 2b - \frac{b_1}{2})t + b_2 \quad (7)$$

The final step is to create a fish shape object around the quadratic curve. Here we used the function $h(t) = 0.284\sqrt{2-t}e^{2t-4}$, see Figure 9. In the code we used 18 points to create the fish to speed up the calculation. Also, we choose more points at the head of the fish ($t \in [1.7, 2]$) because the curvature is higher there.

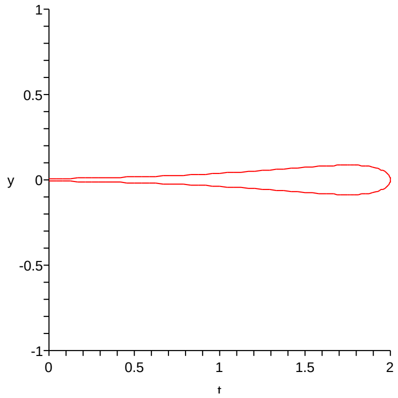


Figure 9: plot of the $\pm 0.284\sqrt{2-t}e^{2t-4}$.

4.2 arrows=curves

The idea of creating the curves is the same as the fishes except that fewer points are needed to compute and render each curve (9 points instead of 18) therefore it is faster to create direction fields with curves. To create a curved shape object, we find the parametric parabola that approximates the solution at the point of calculation. Second we pick nine points on the curve (equally spaced) and draw a thin curve (line thickness 1) through the first six point and a thick curve (line thickness 3) through the last three points. This is a useful option for direction fields with high a number of curves. It is about twice as fast as the fish option. See Figure 10.

5 Animation Options

The `DEplot` command draws the direction field of the system of differential equations at time t_0. With the new animation options, the user can observe where the direction field moves as the time increases. To create the animation of the direction field we used the `dsolve[numeric]` command to find the trajectory of each arrow (at time t_0). Finding the position of each trajectory at the equally spaced time steps we create a series of direction fields for each time step.

5.1 animatefield

`animatefield = true` will create an animation of the direction field with respect to time (i.e. the arrows describing the direction field move as a solution with respect to time). The default number of animation frames is 25 which can be changed with the `numframes` option.

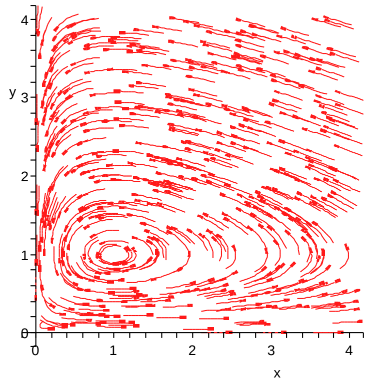


Figure 10: This is a direction field for the Lotka-Volterra model with the use of the `arrows = curve` command. The Maple command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0..2, y=0..2, arrows=curve, dirfield=500);`

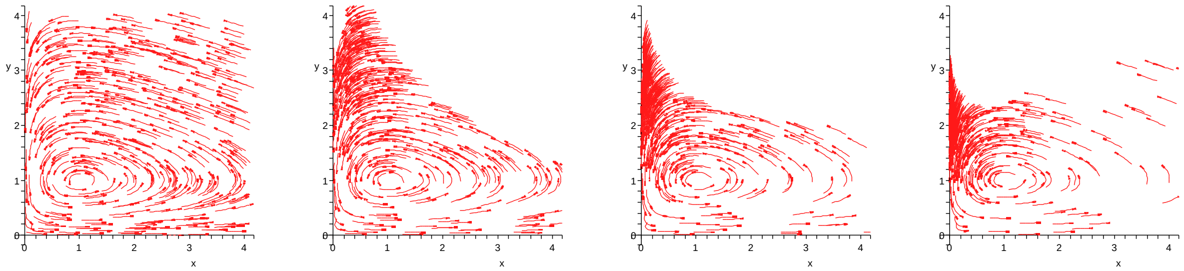


Figure 11: This is a direction field for the Lotka-Volterra model with the use of the `animatefield = true` command. The command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0..2, y=0..2, animatefield=true, dirfield=500, arrows=curve);` From left ro right the frames are for $t = 0$, $t = 1$, $t = 2$, and $t = 3$.

5.2 animatecurve

`animatecurves = true` creates an animation of the solution curve for the specified initial value(s) with respect to time. If `animatefield` is not in use, the number of frames is obtained from `numsteps` (default of 50), but this can be overridden with the `numframes` option. `dsolve [numeric]` is used to compute the solution curves for all specified initial values at different times.

5.3 animate

animate=true is simply a shortcut that can be used to specify that both `animatefield` (if applicable) and `animatecurves` (if applicable) be used. See Figure 13.

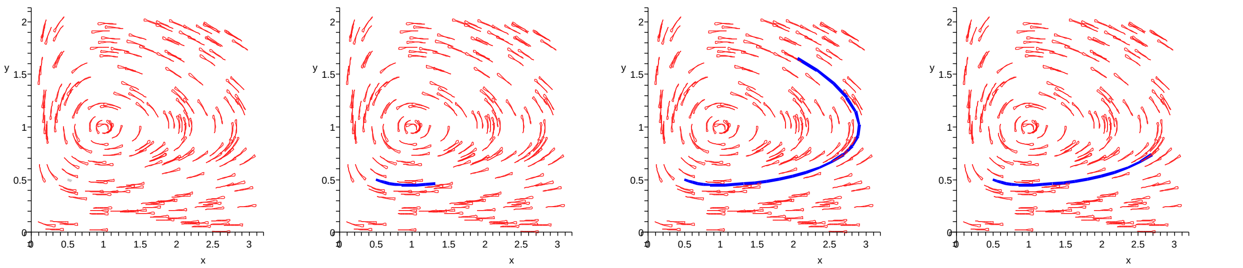


Figure 12: This is a direction field for the Lotka-Volterra model with one solution curve which is animated in time. The Maple command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0.1..3, y=0..2, animatecurve=true, dirfield=200, [[x(0)=0.5, y(0)=0.5]], arrows=fish);` From left to right the frames are for $t = 0$, $t = 2$, $t = 4$, and $t = 6$.

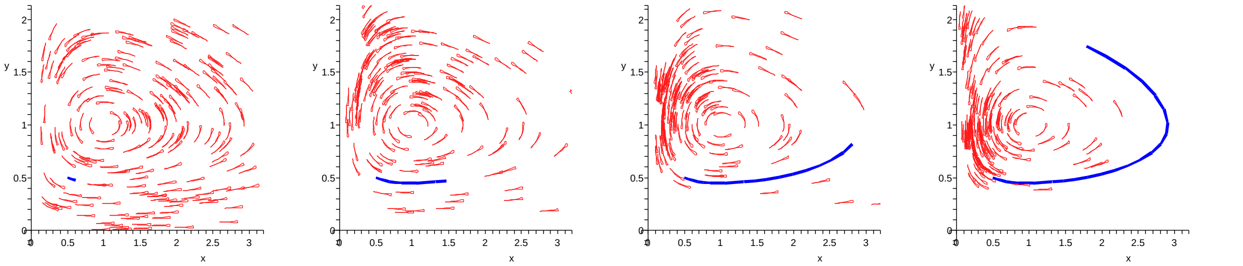
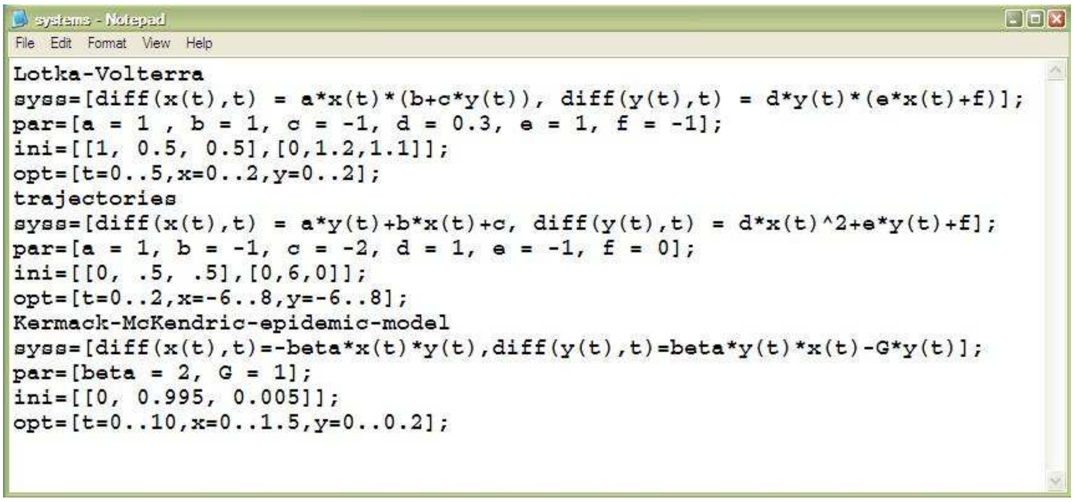


Figure 13: This is a direction field for the Lotka-Volterra model with the use of the `animate = true` option. The Maple command used is: `DEplot(des, [x(t),y(t)], t=1..10, x=0.1..3, y=0..2, animate=true, dirfield=200, [[x(0)=0.5, y(0)=0.5]], arrows=fish);` From left to right the frames are for $t = 0$, $t = 2$, $t = 4$, and $t = 6$.

6 DEplot Interactive, GUI

In addition to the new options for the `DEplot` command we have created a Maplet, `DEplotInteractive`, which can read and write systems of differential equations from and into a text editable library (database) and produce the direction field for the chosen system. Aside from plotting, the user can change the values of parameters, and initial values in the system, as well as some of the options in `DEplot` command. For example, the arrows, range of variables, arrow types, initial values option, and `dirfield` option. The editable database is designed so that it is valid Maple input, thus it is a plain text file, so that the user can read and edit the database easily. The format for the input is described in figure 14 below. Because it is valid Maple input, the Maple user will not need to learn anything as the input format is obvious from the examples in the database. The idea of creating the database is to help educators browse different behaviors of a system of differential equations with different parameter settings and different initial values instantly. This will lead to the idea of having Maple databases of systems, functions and perhaps plots for other commands other than `DEplot` for educators. The database is an alternative way of storing examples than the examples in the Maple's help pages. We think that such databases for other Maple commands and packages should be developed in the future.



```
Lotka-Volterra
syss=[diff(x(t),t) = a*x(t)*(b+c*y(t)), diff(y(t),t) = d*y(t)*(e*x(t)+f)];
par=[a = 1 , b = 1, c = -1, d = 0.3, e = 1, f = -1];
ini=[[1, 0.5, 0.5],[0,1.2,1.1]];
opt=[t=0..5,x=0..2,y=0..2];
trajectories
syss=[diff(x(t),t) = a*y(t)+b*x(t)+c, diff(y(t),t) = d*x(t)^2+e*y(t)+f];
par=[a = 1, b = -1, c = -2, d = 1, e = -1, f = 0];
ini=[[0, .5, .5],[0,6,0]];
opt=[t=0..2,x=-6..8,y=-6..8];
Kermack-McKendric-epidemic-model
syss=[diff(x(t),t)=-beta*x(t)*y(t),diff(y(t),t)=beta*y(t)*x(t)-G*y(t)];
par=[beta = 2, G = 1];
ini=[[0, 0.995, 0.005]];
opt=[t=0..10,x=0..1.5,y=0..0.2];
```

Figure 14: This is a sample of the editable database file. The first line of each system indicates the name of the system. The second line specifies the system. The third line is the list of all parameters used in the system and their default values. The fourth line is the list of initial values to be used, and the fifth line is the names of the variables and their default ranges.

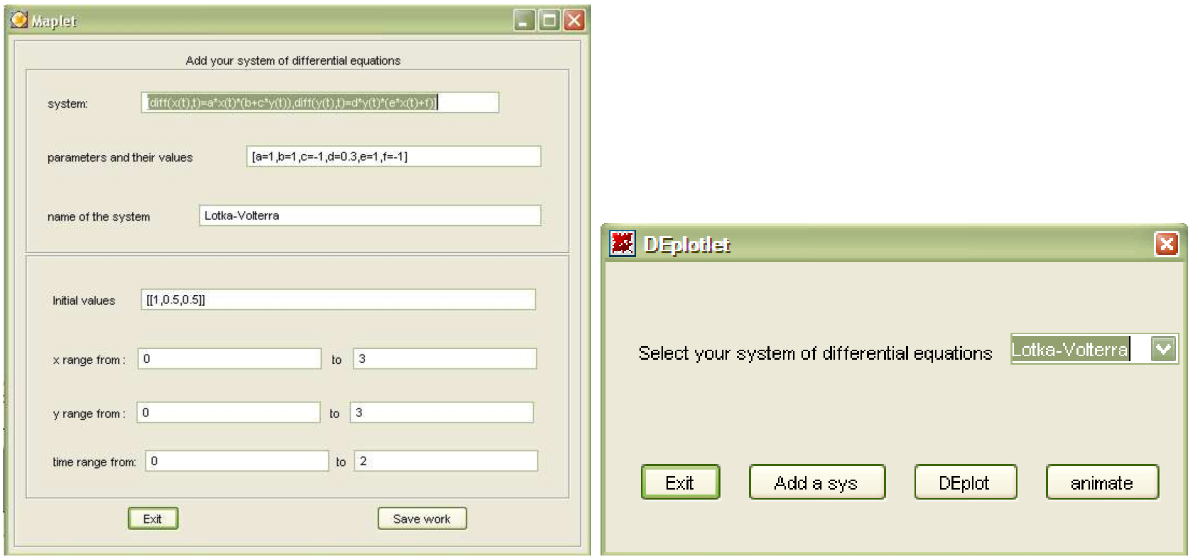


Figure 15: **Left figure:** This is the maplet for adding the system with parameters to the database. The name of the system as well as other information about the system can be specified. **Right figure:** This is the main maplet which user can choose one of the systems, read from the database, to `DEplot` (or animate). In addition, the user can choose to add a new system to the library.

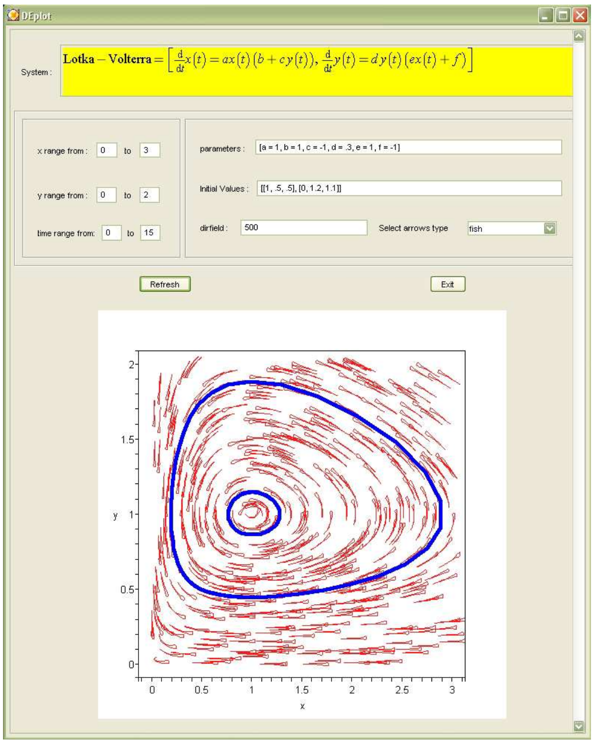


Figure 16: Above is a sample of the `DEplotlet` page. The top window shows the name of the system and the system in a Math-ML view. The two windows below the Math-ML view are for different options of `DEplot` and also values of the parameters and the initial values. These may be changed by the user. Clicking on the refresh button replots the system

7 Acknowledgment

We thank Allan Wittkopf for installing the program, helping with the test file, and his helpful comments. We also thank Robert Corless for his helpful comments.

References

[1] William E.Boyce, Richard C. Diprima. *Elementary Differential Equation and Boundary Value problems.*

[2] Stan Wagon, Dan Schwalbe. *VisualDsolve.* Mathematica Electronic Journal.

[3] M. Monagan, K. Geddes, G. Labahn, S. Vorkoetter, J. McCarra, and P. deMarco. *Maple 9 Advanced programming Guide.* Maplesoft, 2003.