Prefabricated Mathematical Flash Movie Clips

Kevin Cooper
Washington State University
Department of Mathematics
Pullman, WA 99164-3113
kcooper@math.wsu.edu

June 23, 2004

Flash is a Web plugin developed by Macromedia that makes animations within Web browsers comparatively easy to create. The Web is now full of popup menus, twirling bears, and exploding text, thanks to Flash. The power of Flash is to make ”movie clips” move, change size, and rotate without the pain of traditional programming in a language such as Java or Javascript. It is trivial to cause text to change colors, move on and off screen, to make buttons that are associated to actions, and in general to bring life to Web pages without detailed programming knowledge.

The power of Flash for animations gives it great potential for mathematical instructional materials. Indeed, a number of people have already generated extensive libraries of Flash movies to accompany textbooks, and have supervised workshops on the use of Flash in mathematics. These applications are typically specialized for a particular concept that is to be taught, or for an activity to accompany textual materials. Those who have ideas and needs for custom materials for their own courses must develop and program movies for themselves. If the aim is to show balls travelling along parabolas or to animate other objects, this task is not difficult. Paradoxically, standard mathematical tasks such as plotting functions and computing orbits of differential equations are quite difficult in Flash.

This project is really only a demonstration of some ideas - not a completed product. In it, we have a short-term aim, and a long-term goal. The aim of this project is to provide components for those two tasks that may be used by Flash developers to ease their own work in creating instructional materials. In the longer term, we hope that the paradigm presented here might be adopted by others, with a movement toward a creating a library of mathematical instructional movie clips.

This document is intended for Flash developers. If you are not currently familiar with Flash, it should be easy to gain a basic understanding of it using the tutorial materials that come with the package. We estimate that a person familiar with simple programming concepts could learn the basics of Flash in about four hours. We offer no warranties with this estimate.

1 Properties

The components are movie clips for Flash. You may use them and even modify them at will, provided that any derivative code is also in the public domain. We ask also that you provide copies of substantial modifications to us, so that we may make them available to others through the web site. The entire package may be obtained at `http://www.math.wsu.edu/idea/Flash`.

The objects under discussion are *movie clips*. These are small animations that can be used as parts of larger animations. With the appearance of Flash MX such customizable movie clips have come to be called *components*. In this document we will use both terms interchangeably. There are several components in the package.

- **Plot Component:** The main component can plot any number of functions that you specify. It allows you to specify colors and styles, or can revert to default values if you prefer.
- **Point Component:** This can plot a point at coordinates that you specify. Again, you may choose the colors for the point representations. Currently the movie clips use only a disk style of point.
- **Label Components:** There are two label components: one with an arrow indicating the feature it labels, and one without. These can make labels at any points you specify. Thus, you may label points, or simply make text remarks at any location on the plot.
- **Differential Equations:** There is a component that can plot solutions to initial value problems in one or two-dimensional differential equations. You may tell the component whether to plot only forward orbits, backward orbits, or both. You may specify colors for the forward and backward orbits separately.
- **Surface Plots:** There is a component that can create surfaces that form the graphs of single-valued functions of two variables.
- **Parse Component:** This allows the user to put in her own mathematical expressions using a traditional syntax such as that used by Maple.
- **Slide Bars:** These components allows a user to move a slider to control parameters for a plot. They are available in horizontal and vertical formats, with floating point output or integer output.
- **Coordinate Display:** There are components to display the plot coordinates corresponding to the current mouse position.
- **Animators:** These provide a way of varying parameters over time. You specify the parameter or parameters to be changed, the way they are to be changed, and the time over which they are to be changed.

2 Installation

In order to use the package, you must use Microsoft Windows 95, 98, Me, 2000, or XP as your operating system, and you should have Macromedia Flash MX installed. It is no longer sufficient to use Flash 5. We have made changes to the package now that require Flash MX. Since this is a demonstration project, there is no installation package for the components. Download the "Math Components.fla" file from

`http://www.math.wsu.edu/idea/Flash`, and place it in the directory

```
\Program Files\Macromedia\Flash MX\First Run\Components
```

3 Use in Flash MX

To use the movie clips, the basic procedure is as follows.

1. Use Flash MX to open a movie where you want to use the clips.
2. Drag a plotting clip from the library to the movie where it is to be used.
3. Name the clip, for reference.
4. Set variables used by the movie clip.
5. Program the clip with the functions, or differential equations you want to plot.
6. Put other control clips into the movie to allow users to adjust parameters or redraw the plot.

We examine these steps one by one.

3.1 The Components

The movie clips only work in Flash MX. If the installation procedure was successful, then when you open Flash MX you should see a new movie ”stage”. At the right of the screen you should see a list of ”Components”. The components you see should be from the Flash UI Library, but you can change to the Math Components library using the list box at the top of the Components panel.

3.2 Drag a clip

At this point you should see a number of movie clips listed in the Components panel at the right of the screen. Choose one labelled ”Plot (2d Linear Axes)” and drag it to the ”stage” at the center of the screen.

3.3 Name the clip

When you drop the clip on the stage, it becomes your plotting surface. You must name it so that other components can refer to it. To name it, click the box at the lower left of the screen labelled ⟨Instance Name⟩. When you click that box, then it should become blank, and you may type the name of the clip. If you name the clip ”plot”, then other components will already refer to it by default. If you are using more than one plotting surface, then you will have to name at least one of them something other than ”plot”.

3.4 Change parameters

There are many parameters listed at the bottom center of the screen. You may customize your plot by changing the values of these. For the most part, doing so is intuitive, but we provide a list of values for these at the end of this document.

3.5 Program the clip

This is probably the trickiest part. You must write a short program that evaluates the function you want to plot. You can avoid all of this if you choose to use the parser to allow user the user to type the function.

Just above the ”Properties” panel at the bottom of the screen is a gray bar labelled ”Actions”. Click this bar. A panel should open up in the middle of the screen to allow you to enter code. Fortunately, Flash is prepared to help you. At the left side of the ”Actions” panel is a list of categories from which you may choose actions. Click the first blue arrow, labelled ”Actions”. A submenu should appear below it. From that submenu, click ”Movie Clip Control”. A second drop down menu should appear from which you may double-click ”onClipEvent”. In the text box to the right of the menu the following code should appear:

```
onClipEvent () {
}
```

You have an opportunity in the panel at the right to choose the event for which the code applies. Choose ”load”, so that the code now looks like

```
onClipEvent (load) {
}
```

Next, open another submenu of ”Actions”, namely the one called ”User-Defined Functions”. From this submenu double-click the item ”function”. Two text boxes should appear in the panel to the right of the menu. In the first text box you should enter ”fct1”, and in the second ”x”. The first text box is the name of your function, and the second is the argument to that function. The large text entry area at the bottom of the panel should now contain

```
onClipEvent (load) {
 function fct1 (x) {
 }
}
```

Lastly, in the same submenu from which you selected ”function” you should now select ”return”. Again a text box should appear in the center of the panel, in which you enter an expression for the function you want to plot. For example, if you wanted to plot the function $f(x) = x^2$ then you would enter x*x in the text box. More details of the acceptable notation for this box will appear later. At this point the text area at the bottom of the screen should contain

```
onClipEvent (load) {
 function fct1 (x) {
  return x*x;
 }
}
```

You may shrink the ”Actions” panel by clicking its gray title bar again, to reveal the stage.

3.6 Use other control clips

You now have a plotting surface prepared to plot the function you programmed. All you need to do is tell Flash to plot it. The easiest way to do that is to drag a redraw button to the stage. It is best if you put it in

a layer higher than that which contains the plot. To do that, click Insert→Layer from the main menu. Drag the ”Plot Button” clip from the Components panel at the right of the screen to some appropriate place on the stage. The ”Plot Button” both initializes your plot, and allows the user to draw it anew if the need arises.

3.7 Test your movie

To test the movie you have just made, hold the shift key while you press the ⟨Enter⟩ key.

4 Movie Clip Parameters

All of the clips are controlled in similar ways through the parameters that describe them. Some clips have more parameters than others, but there is consistency among them. Here is a summary of the parameters you can set and their functions.

- **refers_to:** This parameter controls the plotting surface that a component applies to. If your plotting surface was named plot, then the default value (_root.plot) does not need to be changed. If you gave your plot a different name, such as ”myplot”, then the value of this variable e.g. for the slide bar component needs to be changed to ”_root.myplot”.
- **parameter_name** This gives the name of the parameter that e.g. a slide bar is to control. The parameter is one that you put into your code for the function to be plotted. Thus if you are plotting the function $x^2 - C$ then you return `x*x-C` from fct1, and you may control the value of C by setting the parameter_name value to ”C”.
- **bottom:** In a slide bar this denotes the lower bound for the function parameter being controlled. In a 2d plot it indicates the lower limit of the y axis display. In a 3d plot it denotes the lower bound of the z axis display.
- **top:** In a slide bar this denotes the upper bound for the parameter being controlled. In a 2d plot it indicates the upper limit of the y axis display. In a 3d plot it is the upper bound on the z axis display.
- **left:** In a plot this indicates the lower limit of the x axis display.
- **right:** In a plot this denotes the upper limit of the x axis display.
- **front:** In a 3d plot this denotes the lower bound of the y axis display.
- **back:** In a 3d plot this denotes the upper bound of the y axis display.
- **label_text:** This is where you can type the text for a label.
- **number_functions:** In a plot this denotes the number of functions to plot. There must be as many functions programmed as this specifies. In other words, if number_functions = 3, then you must have functions named fct1, fct2, and fct3 programmed in the plotting component.
- **border:** This parameter controls the thickness of the border of a plot. A value of 0 means there should be no border, a value of 1 makes a very thin border, and a value of 3 makes a somewhat thin border. There are no other values accepted at this time.

- **background:** This gives the background color for a plot. It can be ”transparent”, or it can be a six-hex-digit color specification such as we are familiar with on the Web. Thus, ff0000 denotes a bright red background, while 006000 denotes a dark green background for the plot.
- **foreground:** This specifies the color for the axes and labels of a plot. It can take the same values as the background parameter.
- **grid:** This variable specifies the color of grid lines. Use ”transparent” for no grid lines, or otherwise specify a color as with the background.
- **npts:** The number of points to use in plotting the graphs of functions. Do not ever set this higher than 250.
- **curvecolor:** This provides access to an array of color specifications for the curves you plot. If the array is empty (the default) then the program picks colors for you. If you want to specify colors yourself, then you must open the array. Select the value box for the parameter, and then click the small magnifying glass at its right (or just double click the box). This opens a dialog box. To add a new color, click the ”+” button at the top left of this dialog. A new box will open in the text area where you can specify a color in the six-hex-digit format used for background and foreground colors. If you specify more than one color, the plotting component cycles through them. In other words, if you plot four functions with two colors, then the first and third graphs will be the first color, and the second and fourth graphs will be the second color.
- **curvestyle:** This is an array to specify the style of the graph. Acceptable choices are ”dash”, ”dot”, or ”solid”. The input is handled similarly to that for the curve color.
- **forward_orbits:** If this is ”true” then the differential equation plotter will plot forward orbits.
- **backward_orbits:** If this is ”true” then the plotter will compute and plot orbits backwards in time.
- **forward_orbit_color:** This is the color for the forward orbits, specified using the usual six-hex-digit format.
- **backward_orbit_color:** This is the color for the backward orbits.
- **orbit_time:** This determines the time over which differential equation solutions will be plotted.
- **duration:** This represents the time that an animation should last, in seconds. In practice, the time may be longer if the computation takes more than 1/12 of a second. The maximum duration is 10 seconds.
- **function_name** This is the name of the function that an expression the user types is to be given. For example, if the user is plotting a function in a display, then the function_name should be `fct1`.

5 Creating Functions

In an earlier section we presented a brief discussion of how to give the function that is to be plotted to Flash. Here we present several more complete examples of acceptable function sets. The function must be typed in a standard computer language syntax. Multiplication is indicated by an asterisk (), and division by a forward slash (/). Those who are familiar with Matlab or Maple might think that exponentiation is indicated using

a carat, but it is not. Instead, Flash provides a built-in exponentiation function called `pow()`. To compute x^3 one would type `Math.pow(x,3)`. Of course, for an integer power like that, it is more efficient to use `x*x*x`, but the pow function may be used for non-integer powers as well, with the usual restrictions on domain. Following is an acceptable script to evaluate x^3.

```
onClipEvent (load) {
 function fct1 (x) {
  return Math.pow(x,3);
 }
}
```

Likewise, the trigonometric, exponential, and log functions also come from the Math object.

We have referred several times to the fact that one can plot more than one function on the same set of axes. All that is necessary to do so is to make more than one function and to tell the plot component how many functions there are. The latter task is accomplished by setting the `number_functions` parameter. The functions are added by iterating the process described in the section above. The following is a well-formed script to plot $x - x^3/3!$ against $\sin x$.

```
onClipEvent (load) {
 function fct1 (x) {
  return x-Math.pow(x,3)/6;
 }
 function fct2 (x) {
  return Math.sin(x);
 }
}
```

We have alluded to the notion that one can introduce parameters to be controlled by the user. Indeed, this is the justification for using a tool such as this. All we need to do to accommodate that use is to use those parameters in our functions. The following is an acceptable function involving two parameters to control amplitude and period of a sinusoidal function.

```
onClipEvent (load) {
 function fct1 (x) {
  return A*Math.sin(B*x);
 }
}
```

Unfortunately, when we view the plot for the above function initially, Flash supposes that A and B are both zero. We need to provide the user with a means of changing them. One way to do this is to drag a slide bar to the stage. We set the "refers_to" variable on the slide bar to the name of the plot component, and we set the parameter_name value to "A". When the user drags the button on the slide bar tool, the parameter A changes. If we drag another slide bar to the stage and set its parameter_name value to "B", then the user can control both the amplitude and period of the sine function.

If one prefers to allow the user to type in values for parameters, then one provides a text box for this. We have not provided a component for this, since it is already easy in Flash. Simply select the text tool from the "Tools" panel in the upper left corner. Before using it, make sure the type of text in the left corner of the "Properties" panel (at the bottom of the screen) is set to "Input Text". Then click anywhere on the stage

to create a text input box. All that remains is to associate the contents of that box with a parameter in the plot component. If we wanted to use this box to set the value of the parameter A in the previous example, we could set the ⟨Instance Name⟩ to "_root.plot.A". If the plot component were named "myplot" instead of "plot", then the instance name would be "_root.myplot.A".

6 Differential Equations

There is a plotting surface designed for handling solution plots and simple phase portraits for systems of differential equations. This is similar to, but somewhat more complicated than the plot component. The critical difference is that one must specify the differential equations whose solutions are to be plotted. The equations *must* be in the form of a two-dimensional first-order system. For example, the differential equation $y' = -\sin(y)$ must first be written in the form

$$\begin{aligned} x' &= 1 \\ y' &= -\sin(y) \end{aligned}$$

For the same reason, the single second-order equation $y'' - cy' + 2y^2 = \cos t$ must be transformed by setting $z = y'$ so that it corresponds to the system

$$\begin{aligned} y' &= z \\ z' &= cz - 2y^2 + \cos t \end{aligned}$$

Once the equations are in this form, it is a simple matter to program them. Following is the program for for the first system.

```
onClipEvent (load) {
 function de1 (x,y,t) {
  return 1;
 }
 function de2 (x,y,t) {
  return -Math.sin(y);
 }
}
```

It is worth noting several differences between this form and that for plotting functions. First, observe that the functions are called `de1` and `de2`, respectively. Secondly, note that these functions take three arguments. The names do not matter, but it is important to understand that the third variable must always be the independent variable for the differential equations. Finally, note that we only return the right-hand sides of the differential equations. Thus, the following is an acceptable rendering of the second-order system discussed earlier.

```
onClipEvent (load) {
 function de1 (y,z,t) {
  return z;
 }
 function de2 (y,z,t) {
  return c*z-2*y*y+Math.cos(t);
```

```
 }
}
```

The differential equation plot component can still plot functions and display points with labels just as the plot component. Thus, one can display solutions and nullclines of the system

$$\begin{aligned} x' &= y \\ y' &= y^2 - x \end{aligned}$$

using the following program.

```
onClipEvent (load) {
 function de1 (x,y,t) {
  return y;
 }
 function de2 (x,y,t) {
  return y*y-x;
 }
 function fct1(x){
  return 0;
 }
 function fct2(x){
  if(x>=0){
    return Math.sqrt(x);
  }
 }
 function fct3(x){
  if(x>=0){
    return -Math.sqrt(x);
  }
 }
}
```

Likewise, one may use the slide bar or a text window to allow users to control parameters in the equations.

There is one other tool that is intended to go with the differential equation plotter. The direction field tool causes vectors indicating the direction of the vector field to be displayed. It does not have any indication of the magnitude of the vectors.

Finally, it is important to note that the when a user starts the Flash movie generated with the differential equation component then she sees only the axes, and perhaps the direction field. In order to see the solution corresponding to a given initial condition, she should left-click the point of the initial condition on the display.

7 3d Surfaces

The package includes a tool to do 3-dimensional surface plots. This tool is in its infancy, and consequently is slow and primitive. On the other hand, it can display the surfaces that form the graphs of functions of two variables in a very small .swf file, and can be used in conjunction with the parser to allow users to type their own functions of two variables to plot.

The component is straightforward in its use. Only a few notes are required. If you choose to program the function for the students, then note that it must take two arguments. For example, the following is an acceptable function.

```
onClipEvent (load) {
 function fct1 (x,y) {
  return Math.sqrt(x*x+y*y);
 }
}
```

While one of the parameters for the component allows you to specify the number of functions to plot, the current version only allows a maximum of two (2) surfaces. For the same reason, you should not try to animate a parameter in the function, or the rotation of the surface - it is just too slow. This will change.

Most of the parameters will be familiar or intuitive. Note, however, that there are new parameters specifying the initial observer position in spherical coordinates. The observer's position is given as a triplet

```
(observer\_distance,observer\_theta,observer\_phi)
```

where the angles theta and phi are the usual angles for a spherical coordinate system, given in degrees.

If you choose to allow the user to type the function, remember to make changes to parameters controlling the parser. In particular, you will have to set the variable array to contain more than one variable (e.g. x and y), and you must make the type of the parser "3dplot".

Slide bars will not work with this component, but for some purposes they would be redundant anyway. In particular, the user can change the viewing angle simply by clicking the plot and dragging the axes around.

8 The Animators

The animator components provide a simple way to vary parameters within plots. The idea is to allow one or several parameters in a component to vary over time in a specified way. For example, if we are plotting the function $x^2 - a$ for different values of a, then we might allow a to vary between 0 and 1 over time t. We could do that in a variety of ways, allowing $a(t) = t$ for t running from 0 to 1, or by setting $a(t) = t^3$, or by setting $a(t) = 1 - t$.

To use an animator, just drag it to the stage. If you only need to change one parameter, drag the "Animator (Single target)" to the stage - it is easier to user. Otherwise, drag the "Animator (Multiple targets)" component. In the "Parameters" panel, set the name of the component in which the parameter you wish to vary resides. Set the "duration" variable to the number of seconds you wish the animation to last. This is a lower bound. If computations take more than 1/12 of a second for each step of the animation, then it will run longer than the number of seconds you specify. Next, for a single target animator, write a function that describes the change of the parameter over time. The function must be called "param". Following is one such function.

```
onClipEvent (load) {
 function param (t) {
  return t/duration;
 }
}
```

This describes a parameter that varies from 0 to 1 over the number of seconds specified. Likewise, to go from 1 to zero one could use the following.

```
onClipEvent (load) {
 function param (t) {
  return 1-t/duration;
 }
}
```

For a multiple target animator, you must enter the names of the variables you need to change in the `parameters` array. Then you must type functions that describe the way those parameters change. The functions are named param1, param2, and so on. An example follows.

```
onClipEvent (load) {
 function param1 (t) {
  return t/duration;
 }
 function param2(t){
  return param1(t)*param1(t)+2;
 }
}
```

When the animation runs, param1 will be used to vary the values of the first parameter in the "parameters" array, param2 will describe the second, and so on.

9 Hints

A few points bear reiteration.

- It is always good to put buttons and other tools that operate on plots in layers that are above the one containing the plot.
- You can animate the motion of a point by an animator for multiple targets: one for the point's `x_coordinate` variable, and one for the `y_coordinate`.
- To get the graphs of functions to be displayed when a movie is loaded, drag a "Plot Button" to a layer above that containing the plot.
- Direction fields are redrawn automatically when a slide bar is used, but the orbits are not. To get new orbits, the user must click a "Clear" button and then choose new initial conditions.

Appendix A: License

This software is distributed under the Gnu Public License. We hope this software will prove useful, but it is distributed without any warranty, without even the implied warranty of merchantability or fitness for a particular purpose. See the GPL below for details.

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an

explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and ”any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM ”AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.