CODEE Journal

Volume 8 Article 1

10-14-2011

Using Technology in a Differential Equations Course: Lessons Learned Implementing a New Paradigm

Thomas Wangler

Using Technology in a Differential Equations Course: Lessons learned Implementing a New Paradigm

Thomas G. Wangler
Benedictine University

Keywords: modeling, software, writing, projects
Manuscript received on June 24, 2011; published on October 14, 2011.

Abstract: Historically, a first course in Ordinary Differential Equations (ODEs) has been taught as a "methods course." Namely, different types of differential equations are trotted out and the method of solution for each class of ODEs is presented. The student is then left to "master the method" by way of lengthy algebraic manipulations, a good dose of calculus, and of course, getting all the pieces of the new method done in the right order. All in all, it can be a daunting task for students and it usually does little to pique the interest or curiosity of most students. This is the old paradigm. In an attempt to make the course more interesting and less daunting, a new paradigm was adopted—a paradigm that incorporates technology as a core component of the course, rather than just an "add on." This article examines various issues related to implementing this new paradigm as well as some lessons learned along the way.

1 Background

Differential Equations is a four credit hour course in the mathematics curriculum at Benedictine University. Students usually take it after taking multivariate calculus and before (or along with) Linear Algebra. Typically, about 25% of the students are mathematics majors and about 75% are majoring in the natural sciences or engineering. The former group of students takes the course because it's required for all mathematics majors and the latter group because it's a cognate requirement for their major or because they are working on a minor in mathematics. Differential Equations is designated as a Writing Intensive (WI) course. This will be discussed in more detail later in the paper.

Since Differential Equations is a transition course from lower level (more computationally oriented) courses to upper level (more theoretically oriented) courses, there are two main learning objectives. The first is to impart knowledge and understanding of the course content. This includes the methods required to obtain solutions for various classes of ODEs or a system of ODEs and how to analyze an ODE in order to make a qualitative statement about the solution without knowing what it is, e.g. use the phase portrait to discuss the stability of equilibrium solutions. The second is to foster growth in

each student toward the next level of mathematical thinking. This involves things like modeling a physical phenomenon mathematically, analyzing a model, bringing theoretical questions to bear on a model and using theorems to answer these questions, and learning how to think more abstractly about mathematics. The idea is to strike the right balance between the theoretical and the applied nature of the material and to use technology in some meaningful way to help the students learn the mathematics. Therein lies the challenge!

2 Technology Platforms

The main technology platform used in the course is Maple, a computer algebra system. Other pieces of technology used include

1. the Desire2Learn (D2L) course management system,
2. PowerPoint slides, which come with the text and are uploaded to D2L for ease of access,
3. Maple files, which also come with the text and are related to HW problems or textbook examples,
4. the web, in the form of links to related web sites (these are posted to D2L); websites used for in-class demos, e.g. YouTube videos on bungee jumping; and the student companion website, which comes with the text and has many additional resources for the students, like chapter summaries and interesting projects that go beyond the course content.

All in all, Differential Equations is a technology rich course.

Before discussing how Maple is used in the course, a brief summary of the four software packages used over the past 21 years will first be given. The first piece of software used (1993) was MODELS. This software is user-friendly, has some very nice graphing and animating capabilities (it graphs in 2D and 3D), and allows the user to specify which numerical solver is used. The pro for MODELS was that it allows for easy creation and modification of lab explorations. The con was that you had to create the labs *ex nihilo*.

The second piece of software used (1998) was Interactive Differential Equations (IDE). This software runs in a web browser and comes with a workbook that has lab experiments for students to work through and interactive illustrations that are useful for classroom demonstrations and discussions. The pro for IDE was that it came with pre-made lab experiments—a huge time saver! The con was the lab experiments couldn't be (easily) modified, so the students were doing the same ones over and over.

Although IDE had no glaring deficiencies, when ODE Architect hit the streets (around 1999), the IDE software was retired because ODE Architect did everything IDE did . . . and more. For instance, the videos that come with the ODE Architect software are great "attention grabbers" at the beginning of class since they show students how differential equations can be used to model problems/phenomena in the real-world. The multimedia piece of ODE Architect is useful for in-class demonstrations and to "prime the pump" for explorations in the computer lab. ODE Architect also has a handy interface with Windows Word Pad by way of the Report Writer tool on the Toolbar. This allows students

to copy-and-paste graphs and equations seamlessly into Word Pad, so students are able to complete their lab reports without needing any "outside" software. The software's graphing capabilities give the user a wide range of options and—as with the MODELS software—the user can pick which algorithm is used to obtain the numerical solution. As if this isn't enough, there is a companion workbook by Borrelli and Coleman [1] containing a wide variety of lab experiments, which can be easily adapted and put into a lab report format.

In light of all this, why would anyone move away from the ODE Architect software? There are two reasons that caused the author to switch from ODE Architect to Maple. First, ODE Architect is a 16-bit application with compatibility with 32-bit operating systems. In the spring of 2010, several students were not able to download and run ODE Architect (from the student companion website) because their laptops were running 64-bit operating systems (either Windows Vista or Windows 7). In the process of resolving this "bit issue," the author was informed by technical support staff at John Wiley & Sons (January, 2010) that there are no plans to upgrade ODE Architect to be compatible with a 64-bit system. Second, Maple is used throughout the three-course calculus sequence at Benedictine University, so by the time students get to Differential Equations, most of them are already quite familiar with Maple. This being the case, the questions comes knocking, "Why not use a piece of software the students already know rather than requiring them to learn yet another piece of software?" So, due to the bit incompatibility issue and the desire to keep the number of software packages used by students to a minimum, the author (reluctantly) switched to Maple in the spring of 2011.

3 Implementing the New Paradigm

To implement technology as a core component of the course and not merely as an"add on," the class meets in a computer lab five full class periods (out of 28) throughout the semester. Put another way, the author has to "give up" about 18% of his class time (=lecture time) to effect the paradigm shift. Given that the labs constitute 19% of the students' overall grade, this seems like a reasonable price to pay to get technology into the course. Here's how the labs are set up.

The lab is written and delivered to the students (via D2L) in a Maple file, which has two main sections. The discussion section comes first and goes over the physical context of the lab and any background information the students need, the modeling process, the math behind the model, and the Maple commands needed to complete the lab report. It typically takes about half the period (50 minutes) to complete this section. During the second half of the class period, students work through the other section of the lab, which is the lab report section. See Appendix I and II for examples of two labs. They can do this on their own or with a lab partner, but they are encouraged to work with someone else as they complete their lab report. During this time, the instructor is available to the students for help and proactively checks on them to gauge how they're doing. All this is done in a computer lab equipped with 20 desktop computers.

The lab reports are not designed to be completed in two hours so students finish them up outside of class. On average it takes a student about 5–6 hours to complete a lab

report and they're given one to two weeks to do it, depending on what else is going on in class. Students complete about five labs during the semester. This seems to be about the right number of labs, especially in light of devoting five class periods to going with the students into the computer lab to get them started on the right track (as opposed to simply giving them a lab assignment and telling them to "use Maple to do it"). Based on students' comments over the years, somewhere between four and six labs has worked well.

Earlier in this article, it was mentioned that Differential Equations is a Writing Intensive (WI) class. A few comments are in order at this point. The Writing Program at Benedictine University subscribes to the Writing Across the Curriculum philosophy, namely, that writing is central to students' education and future success and belongs in every discipline. Among other things, the program requires students to complete at least one WI course in their major. WI courses are defined as not only requiring substantial writing, but as providing instruction in writing as well. One of the distinctive features of WI courses is a focus on writing process, including at least one draft critique by either instructor or peers.

Since Differential Equations is a WI course, students are required to answer all questions in complete English sentences. (This applies to all lab reports and some homework problems, but not to in-class tests.) The British proverb applies here: "What one cannot clearly state, one does not really know." Math is no exception! So, giving students labs is one way of getting them to write about mathematics, and this can make it clear what a student knows (or doesn't know) and that can help the instructor know where more time needs to be spent. Also, based on student comments over the last five years, students find the peer-review process for lab reports quite helpful.

Here's how the peer-review process works. Before students turn in their final draft, they give it to another student to review. This is usually done in the computer lab at the beginning of the period. Since each student has completed his/her lab at this point, they are able to go over each others' labs and give meaningful feedback. Once they're done with the peer-review process, labs are given back and some time is allowed for students to ask their peer-reviewer any questions they have about comments written on their lab report. Once that is done, students have two days to make changes before turning in the final draft of their lab report. Incidentally, students get five points (out of 55) for performing this peer-review function for their classmates—*if* they have already completed their own lab. The students who do not have their lab completed at the time of peer-review forfeit these five points, the idea being they are not able to review someone else's lab if they haven't done their own.

Long before students get to the peer-review process, they need guidance regarding what is expected of them in writing a lab report. An important aspect of this guidance, possibly the most important, is the collaboration policy that is in effect for the labs. The following collaboration policy is given to the students the first time they go into the computer lab: "The math department encourages students to work together on lab reports. Since the rules for student collaboration vary from department to department, the following guidance is provided to clarify what is allowed when working with fellow students on lab reports. You are encouraged to collaborate with a classmate as you work through the lab, but once you have solved the problem and are ready to *write-it-up-in-a-text-box*, that's where you must part company. I fully expect your equations and graphs

to be the same as your lab partners, but *when you answer the lab questions in a text box, that work must be your own work*, and hence, I expect it to be significantly different than your lab partner's."

In addition to the collaboration policy, the first time students go into the computer lab they are given the following guidance for writing a lab report:

- *Every* graph deserves an interpretation and/or description. Do *not* simply graph something and then make no comment about it. There is always a reason for graphing and the graphs mean something or tell you something. Tell *me* what the graph tells *you*.
- When you are asked to do some exploration and then draw a conclusion based on the results, be sure to explain why you drew the conclusion you did. Don't just state the conclusion with no explanation and/or no graphs.
- Label your graphs appropriately, especially when you are doing a sweep over different parameter values. For example, in the case of constant effort harvesting you will have to sweep over several values of h and Maple produces one curve for each value. You should label which curve corresponds to which value of h. You can do this by annotating the graph.
- The conclusions you draw, based on a numerical investigation, should be consistent with your graph. It is more important that you draw reasonable conclusions based on a graph, than that you make a perfectly precise statement that is technically correct all the time.
- Interpret the graphs and answer the questions *in light of the context*. So, instead of saying "the graph approaches zero," say, "the population of sardines is driven to extinction."
- When told to "Try different values for *IC* and . . ." you must do some numerical work and include graphs in your report. In other words, don't just do some work and then answer the question without including graphs in your report. Any statement you make in the report that is based on a graph should have a graph in the report for me to look at. Also, if you do three or four explorations and they all yield the same result/conclusion, you can include a "representative graph" instead of all four.
- When asked to do "stuff" and *summarize your results*, you need to include graphs and/or equations and a few sentences that summarize what the results are telling you.
- Answer the question that is asked. Don't answer a different question. Be careful here! It is very easy to answer a question that is "close but different" than the one asked. I suggest that you read and re-read the question.
- Answer all questions in complete sentences. This is the "Writing Across the Curriculum" aspect of lab reports. Be clear. Be concise. Be complete. Be careful. But don't be verbose! I do *not* give extra points for waxing eloquent.
- General rule: When in doubt, put it in the lab report. You will never lose points for having extra "stuff" in your report, but you will lose points when pieces of the report are missing in action.

- I'll end with something you need to pay attention to so it doesn't come back to bite you! The full lab report is due on the day we do the peer-review. Each peer-reviewer will be required to annotate the lab he/she reviews to indicate how much of the lab was completed on that day. If your lab is not complete on the day of the peer-review, you will forfeit those points. (You will be turning in both your peer-reviewed copy and the final copy of your lab report to me.) Don't get bit by this! Pace yourself. Manage your time carefully. And get the lab done on time.

The last thing students are given is feedback from the instructor after grading the first lab. This is usually extensive feedback on where the students lost points and how they should have done the lab, i.e. what the instructor was expecting of them. So, in the overall lab report process, students are given guidance by the instructor before they do their first lab, input from their classmates on their first draft (by way of the peer-review), and feedback from the instructor after the lab is graded.

It should be noted that, although the lab component is the most Maple-intensive piece of the course, Maple is used in other ways as well. In fact, Maple is used virtually every week by the instructor or the students or both, in one way or another. For instance, Maple is used for in-class demonstrations like first solving a system of ODEs by hand, and then using Maple to plot the direction field and graph the solution curve so students can see what the solution looks like. Alternatively, Maple might be used to show the students an animation or to illustrate how the behavior of the solution changes as we change a parameter value.

Sometimes Maple worksheets with graphs and questions are handed out to students to work on in groups, discuss their answers with one another, and then report to the entire class. Other times it might be used simply to expedite working through an example. For instance, after finding one eigenvalue/eigenvector pair for a 3×3 matrix—by hand—the students are shown how to use Maple to find the other two eigenvalue/eigenvector pairs and then all this information is used to write down the final solution. So, using Maple saves time. Whether Maple is used as an in-class demo, for group work, or to save time delivering content, the Maple files are always made available to students to use for homework or review as needed.

Regarding homework, students are assigned some homework exercises that require the use of Maple—usually to plot a direction field or phase portrait or to graph a solution and then answer questions based on the graph. So, over the course of a semester, students use Maple in the computer lab, see it used in the class room (on a regular basis), and use it for their homework exercises (as needed). One might reasonably ask whether students have access to Maple while taking in-class tests. Rather than giving the students access to Maple during a test, the author has taken the approach of using Maple to make graphs and direction fields himself, pasting them into the test, and then asking questions about them or requiring students to describe, explain, or infer something based on the graph or direction field.

4 Why Bother Using Technology?

Some people tend to view the use of technology in math classes as the panacea we've all been waiting for and yet, if a piece of software or technology is not used properly, it

can amount to little more than a high-tech pedagogical placebo. It is important to keep in mind that technology (in math classes) is a means to an end and not an end in itself. The goal is to impart knowledge and understanding and develop critical thinking skills in mathematics. To the degree that technology helps an instructor teach his/her students mathematics, it is a useful tool and this instructor has found the use of technology in ODEs (in the form of a computer lab component) to be worth the effort for the following reasons.

- It ensures a minimal level of computer competency and hence contributes to the program's and/or university's student learning objectives with respect to the use of appropriate technology in the student's field of study.
- It facilitates exploration and discovery, e.g. vary the harvesting parameter over the following range to see what happens . . .
- It helps students think more abstractly and critically because they are able to focus on the mathematical concepts rather than lengthy calculations or algebraic manipulations.
- It allows students to "see the mathematics" through graphs and animations. This in turn oftentimes helps students understand the mathematics better.
- It improves students' writing skills by requiring them to write lab reports in complete English sentences, using proper grammar and punctuation.
- Students generally like the computer labs, some going as far as citing the labs as "the greatest asset" or "brightest spot" of the class.

The above list gives the pros of implementing a computer lab experience into an ODE class. A fair question is whether there are any cons associated with this. Put another way, "What's so hard about implementing technology into a math class?" Two things should be mentioned in response to this question.

First, there is a significant time commitment, especially the first time through the process. For instance, it takes time to provide the students with the guidance they need regarding the software itself as well as the instructor's expectations for lab reports. Then there are the inevitable system issues (e.g. printer problems), software issues (e.g. software won't run on a student's computer), and computer issues (e.g. computer freezes up midway through the lab). The biggest time commitment is the time required for creating, delivering, grading, and improving the labs themselves. Second, creating a "good" lab can be challenging because it should be interesting; the instructions to the students should be clear and easy to follow; it is desirable to have an element of exploration and discovery; it should lead students to a deeper understanding of the mathematical concepts; the questions should be doable but not trivial; and students should not be required to spend an inordinate amount of time to complete their lab report.

In spite of the prodigious time commitment and the very real challenge of creating good lab explorations, 18 years of using technology has convinced the author that the pros far outweigh the cons.

5 Resources for Writing a Lab

So how does one go about writing a lab in an ODE course? There are a number of resources available to instructors that can result in a computer lab experiment for students. A rich resource, mentioned earlier in this article, is Borrelli and Coleman [1], which this author has used to compose several labs. The web offers a host of resources, such as

1. Internet Differential Equations Activities (IDEA), which is an interdisciplinary effort to provide students and teachers with computer based activities for differential equations in a wide variety of disciplines. The IDEA homepage is at www.sci.wsu.edu/idea and an interesting exploration involving bungee jumping is at www.sci.wsu.edu/idea/Bungee.

2. The Community of ODE Educators (CODEE) Digital Library is an online repository of materials related to teaching and learning ODEs and is available at www.codee.org.

3. In addition to these web resources, a Google search on "differential equations experiments" will produce several thousand sites.

Textbooks will oftentimes have projects or extended homework problems that can be converted into a lab experiment. For instance, this author has used problems from Boyce and DiPrima [2, page 131] to create a lab on cobwebs and chaos and [2, page 90] to create a lab on logistic growth with harvesting. Additionally, [2] has an instructor's companion site associated with it that has projects which are suitable for converting to lab explorations. Finally, journal articles can be a fruitful resource on ideas for ODE labs. For instance, this author used an article by Steven Strogatz [3] to create a lab modeling the love affair between Romeo and Juliet—definitely the lighter side of mathematics and invariably the students' favorite lab!

6 Lab Explorations and Students' Reactions

As an example of the kinds of labs one might create, the list below is offered with only brief parenthetical comments. It should be noted that in any given term, one might assign between four and six of these labs. For instance, in the most recent term, the first four labs were used with the fifth lab being scrapped due to . . . a snow day in Chicago.

- Exploring Different Population Models (introduction to Maple via population models). (See Appendix I.)
- Slope Fields and Skydiving (falling body problem with linear drag). (See Appendix II.)
- Logistic Growth with Harvesting (an exploration of the logistic ODE with various harvesting assumptions).
- Romeo and Juliet (a system of ODEs modeling a love affair that leads to complex eigenvalues).
- Cobwebs and Chaos (an investigation of the discrete logistic equation leading to chaos).

- Springs and Oscillations (an exploration of the impact of physical parameters on a mass-spring system).
- Newton's Law of Cooling (mathematical modeling and fitting a curve to data).
- Pizza and Video (a system of linear ODEs modeling seasonal fluctuations in sales at two nearby stores).
- Predator/Prey Models (an autonomous system of nonlinear ODEs modeling the inter-species dynamics of a predator and its prey).

What do the students say about the lab component of the class? As one might expect, some think it's the greatest asset of the class, some don't particularly care for it (usually those who don't like computers) and some are neutral. A sampling of student comments about MODELS, ODE Architect, and Maple is given below.

- MODELS was a great asset to the class. It was very easy to use and really helped me to visualize what was going on.
- I liked MODELS more than I thought I would. After you get the hang of it, it's very easy to work with and produce excellent graphs.
- I thought the MODELS labs were interesting and provided a different way of looking at the material.
- MODELS brought the material to life and made the mathematics more understandable.
- I liked ODE Architect. It helped me relate the material to real-world problems, which helped boost my interest.
- ODE Architect is a nice tool, though sometimes difficult to use. I thought it was used very effectively.
- ODE Architect was very helpful but they need to fix the bug so it works on 64-bit computers. I had to use a school computer to do all my labs because I couldn't install ODE Architect on my laptop. ☹
- I liked the labs. Some were a little long, but the real-life application helped. ODE Architect was great for visualizing what's going on.
- I liked the way we used ODE Architect. Once I got the feel for the program, it was helpful and useful. I know the labs certainly helped my performance in the class.
- ODE Architect helped me learn the material. It clarified topics covered in class and it showed applications of ODEs.
- I found Maple really helpful in modeling and applications of ODEs in the real world.
- I kind of liked the Maple labs, but it took a long time to do them.
- Maple was useful and I think it was used effectively.
- I didn't like the Maple labs because you had to input things in a certain order. As a side note though, if the labs continue, don't change them. I liked the real-world examples.
- I definitely learned a lot of new things from Maple.

These comments suggest that the particular piece of software used is of secondary importance because students say the same kinds of things about different software packages. Namely, that the labs make the math more interesting, they really like the real-life

applications, and being able to visualize what's going on (through graphs, direction fields, phase portraits, and animations) helps them to understand the math better. The moral of the story seems to be "pick your software; use it well!"

7 Conclusion

Implementing a technology component into a Differential Equations class (or any math class for that matter) is a worthwhile undertaking. It takes a significant amount of thought, time, and effort on the part of the instructor, but if done right, it can pique the interest of students, show them real-world applications of math, and help them understand the math we are trying to teach them. In short, using the technology can help our students learn the mathematics—and that's a good thing!

References

[1] Robert L. Borrelli and Courtney Coleman. *ODE Architect Companion.* J. Wiley and Sons, 1999.

[2] William Boyce and Richard DiPrima. *Elementary Differential Equations.* Wiley, 2009.

[3] Steven Strogatz. Love affairs and differential equations. *Mathematics Magazine*, February 1988.

Appendix I: Lab 1 - Harvesting a Biomass

Discussion for Lab 1

1. At the beginning of each lab, you will include the following three commands:

```
> restart
> with(DEtools) :
> with(plots) :
```

These three commands are needed for the kinds of things you'll be doing in DiffyQ labs. Maple will not give any output for these commands because of the ':' at the end, which suppresses output.

2. The most basic command (for DiffyQ labs) is the command that authors and names an ODE. You always author commands on a command line, which is a line that starts with [>. Here's an example.

```
> ode1 := diff(y(t), t) = k·y(t)·(y(t) − 8)
```

$$ode1 := \frac{d}{dt} y(t) = k\,y(t)\,(y(t) - 8) \tag{1.1}$$

Now, any time you type *ode1* in a command line, Maple knows your talking about the ODE given above. Go ahead and type *ode1* in the command line below.

```
> ode1
```

$$\frac{d}{dt} y(t) = k\,y(t)\,(y(t) - 8) \tag{1.2}$$

Let's talk about some Typesetting commands. Notice that $diff(y(t), t)$ is represented by Maple as $\frac{d}{dt} y(t)$. To get Maple to use the prime notation use the following two commands:

```
> interface(typesetting = extended) :
> Typesetting[Settings](useprime, prime = t)
```

$$true, x \tag{1.3}$$

Now check what the ODE looks like by typing *ode1* in a command line.

```
> ode1
```

$$y'(t) = k\,y(t)\,(y(t) - 8) \tag{1.4}$$

Oftentimes, we'd like to suppress the t dependence. In other words, we'd like to work with y instead of $y(t)$. To do this, we use the following command:

```
> Typesetting[Suppress](y(t))
```

Again, check what the ODE looks like by typing *ode1* in a command line.

```
> ode1
```

$$y' = k\,y\,(y - 8) \tag{1.5}$$

3. Assigning (and unassigning) values to a parameter. Let's keep working with *ode1*. Suppose we wish to assign the value of -0.8 to the parameter k in *ode1*.

```
> k := -0.8
```

$$k := -0.8 \tag{1.6}$$

Now check.

```
> ode1
```

$$y' = -0.8\,y\,(y - 8) \tag{1.7}$$

Note that the form of the Maple output is the same as the form of our input. In other words, we assigned a value to k that was in decimal form and Maple represented it in decimal form. What if we had assigned a value to k in rational form?

> $k := -\frac{8}{10}$

$$k := -\frac{4}{5} \tag{1.8}$$

> *ode1*

$$y' = -\frac{4\,y\,(y-8)}{5} \tag{1.9}$$

The moral of the story is that Maple always adjusts its output to match our input. This comes in handy sometimes. Now, what about unassigning a value to a parameter. If you want to clear the current value and reset the parameter as an "arbitrary constant" with no value assigned to it, you do as follows:

> *unassign*('*k*')

Now check.

> *ode1*

$$y' = k\,y\,(y-8) \tag{1.10}$$

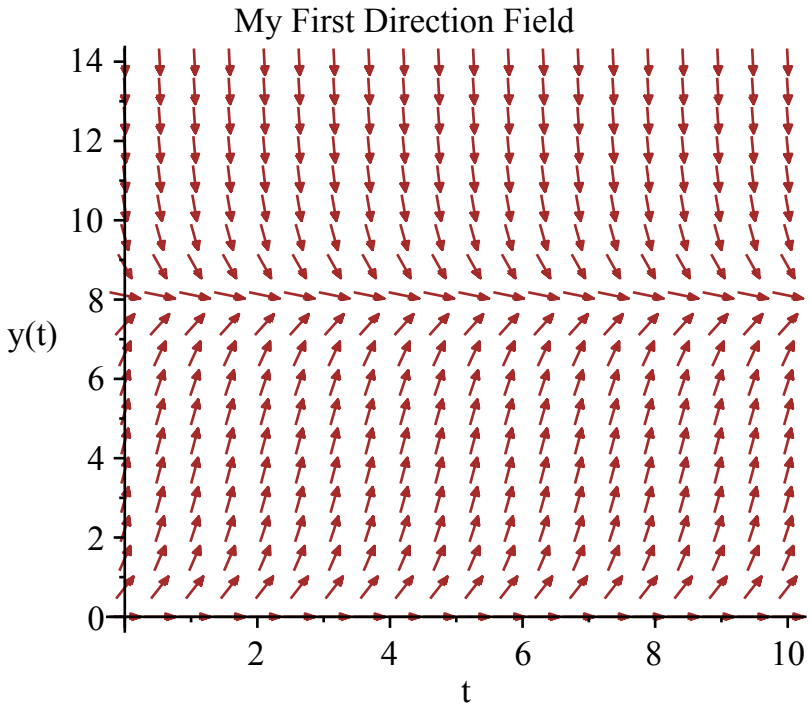
So we see that *k* is no longer -8/10, but has been set back to its original unassigned state of *k*.

4. Let's now consider how to solve an ODE and plot the solution. There are two commands that we will use over and over in our DiffyQ labs. Here's the first one. (Before we plot, we must reassign a value to *k*.)

> $k := -0.4$

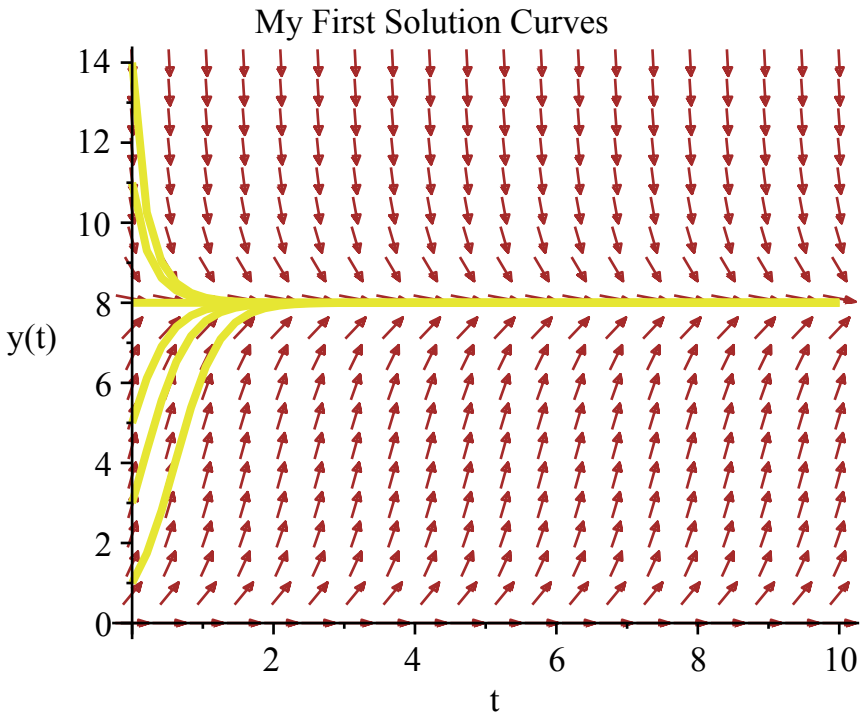
$$k := -0.4 \tag{1.11}$$

> *DEplot*(*ode1*, $y(t)$, $t = 0..10$, $y = 0..14$, *arrows* = *medium*, *color* = *brown*, *title* = "My First Direction Field")



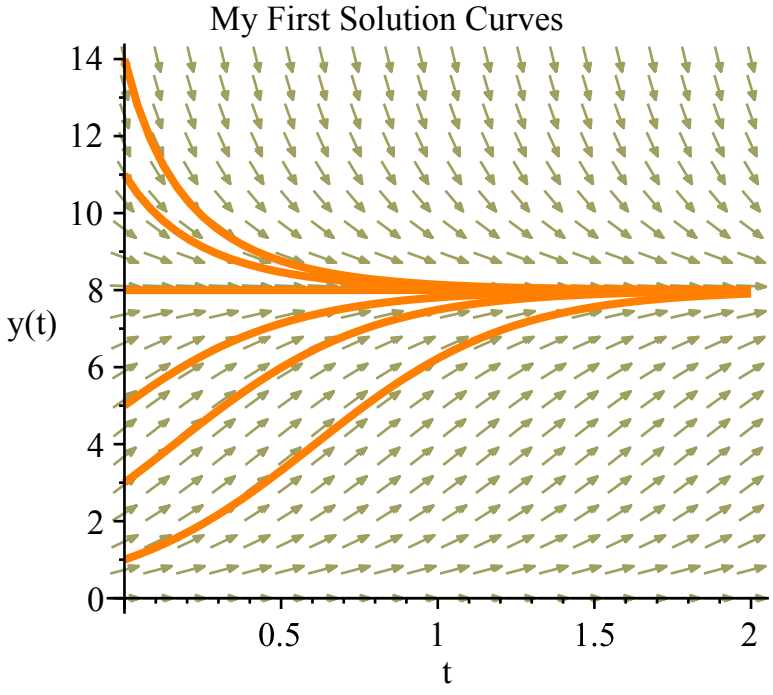
From the above direction field, it looks like the solutions are approaching $y = 8$ as $t \to \infty$. We can insert some solutions to confirm or refute this. To insert solutions into the direction field for various Initial Conditions (ICs), you simply list each IC using brackets [,] separated by semicolons *and put all the ICs listed within a set of brackets.* See below command.

```
> DEplot(ode1, y(t), t = 0 ..10, y = 0 ..14, [[0, 1], [0, 3], [0, 5], [0, 8], [0, 11], [0, 14]],
      arrows = medium, color = brown, title = "My First Solution Curves")
```



The default color for solutions is yellow, which doesn't show up well in a printout, so you will always have to change that. Also, from the above graph, we can see that all the space from $t = 2$ to $t = 10$ is wasted space on the graph because nothing's really happening. So, we should "stretch out" the graph by letting time vary from 0 to 2. We will change the color and the scaling in one shot. Here goes.

```
> DEplot(ode1, y(t), t = 0 ..2, y = 0 ..14, [[0, 1], [0, 3], [0, 5], [0, 8], [0, 11], [0, 14]], arrows
      = medium, color = khaki, linecolor = coral, title = "My First Solution Curves")
```

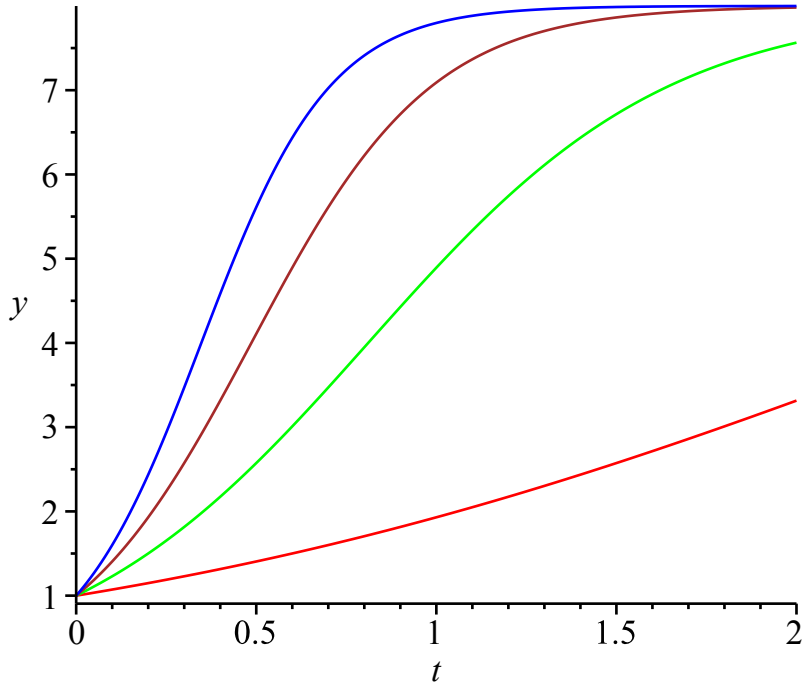


Notice that I chose a different color for the arrows in the direction field as well as a different color for the solution curves. You can choose many differnt colors in Maple. Keep in mind that you want to choose colors that will show up when you get a printout. Other than that, you can choose whatever color you wish. For a list of all available colors in Maple go to Maple Help, then to the Table of Contents tab, then Graphics, and finally, Plot Color Names.

5. The other very useful command is the one we use when we keep the IC the same and allow some parameter to vary. Let's suppose we wish to let the parameter k vary over some values and keep the IC at $y(0) = 1$ in order to investigate the affect of the parameter k on the solutions. Here is what we'd do.

```
> sol1 := dsolve([diff(y(t),t) = -0.1*y(t)*(y(t)-8), y(0) = 1], numeric);
  sol2 := dsolve([diff(y(t),t) = -0.3*y(t)*(y(t)-8), y(0) = 1], numeric);
  sol3 := dsolve([diff(y(t),t) = -0.5*y(t)*(y(t)-8), y(0) = 1], numeric);
  sol4 := dsolve([diff(y(t),t) = -0.7*y(t)*(y(t)-8), y(0) = 1], numeric);
  p1 := plots[odeplot](sol1, 0..2, color = red); p2 := plots[odeplot](sol2, 0..2, color
      = green);
  p3 := plots[odeplot](sol3, 0..2, color = brown);
  p4 := plots[odeplot](sol4, 0..2, color = blue); display([p1, p2, p3, p4]);
```

$$sol1 := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$
$$sol2 := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$
$$sol3 := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$
$$sol4 := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$
$$p1 := PLOT(\ldots)$$
$$p2 := PLOT(\ldots)$$
$$p3 := PLOT(\ldots)$$
$$p4 := PLOT(\ldots)$$

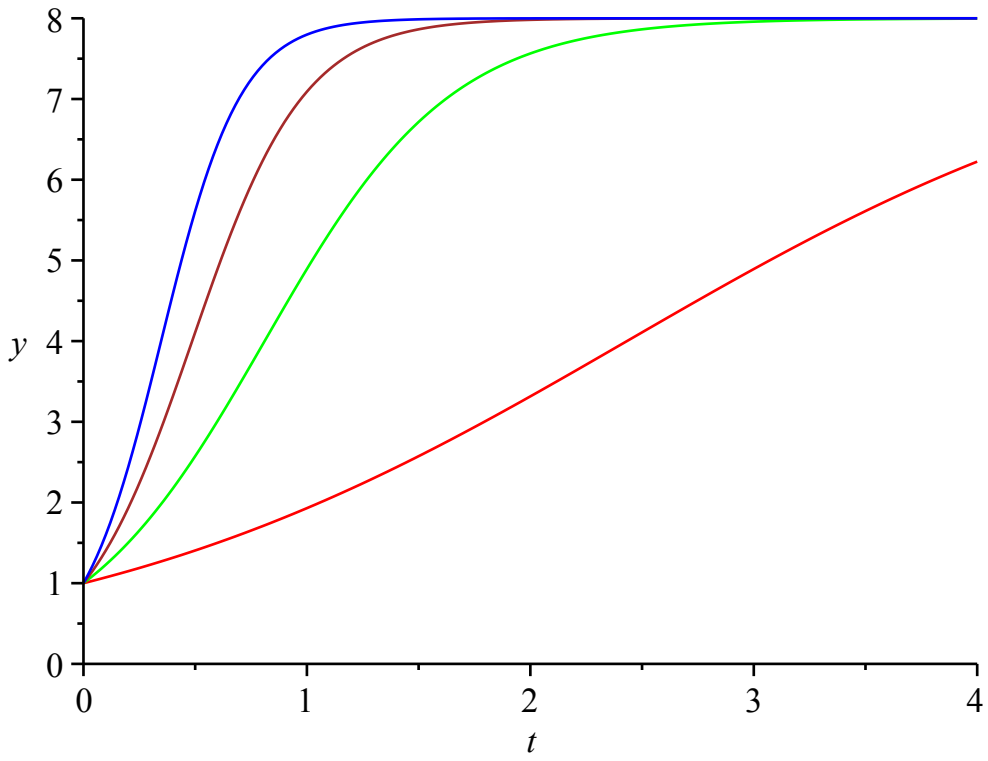


If the graph does't display, you may have to re-execute the *with(plots):* command first and then execute the above set of commands.
You need to notice a few things here:
Thing1 - We had to use the *dsolve* command one time for each value of the parameter, using the

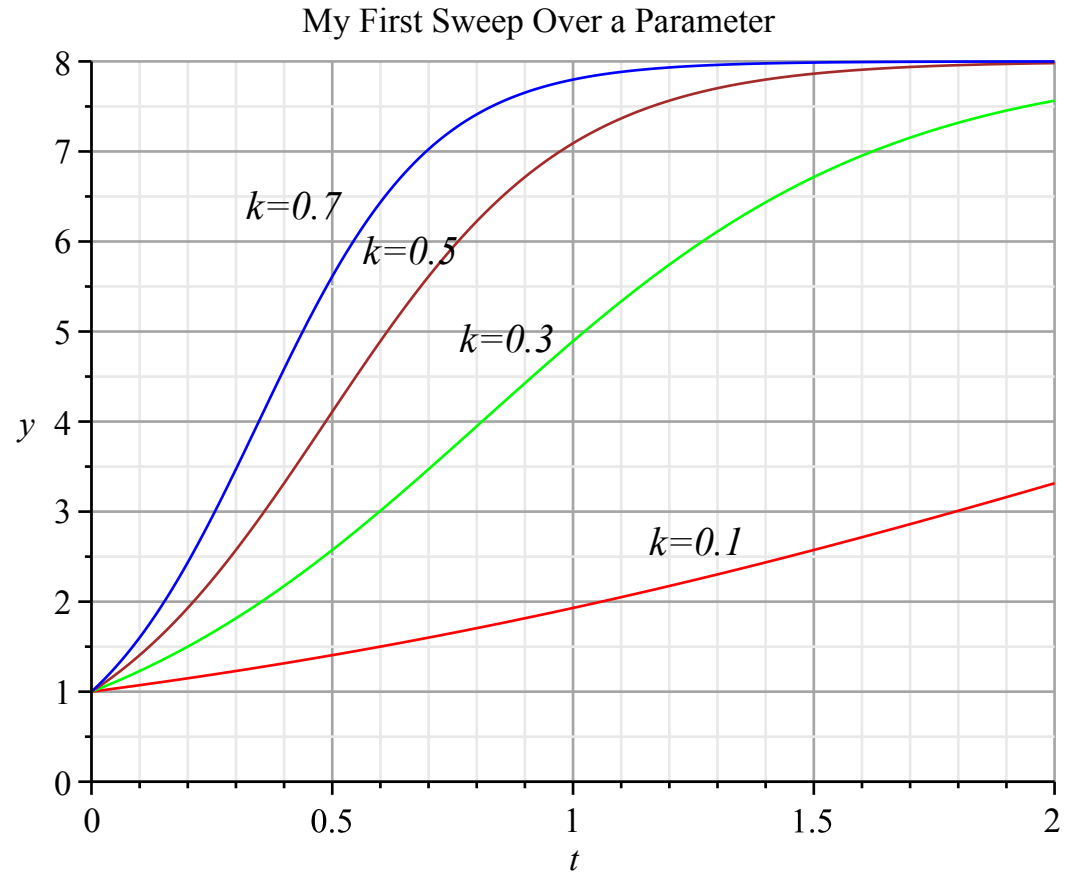
same IC each time.
Thing2 - We had to type out the ODE (instead of using *ode1*) because we needed to insert different values of *k* into the ODE.
Thing3 - We assigned a name to each solution as we obtained it (*sol1, sol2, ...*) and then plotted each one (giving each a different color but suppressing the plot by assigning each plot a name, *p1, p2, ...*), and finally, we used the *display* command to display all the plots. Before using the *display command*, you must execute the *with(plots):* command, or it won't work! This is why we will always execute the *with(plots):* command at the very beginning of our DiffyQ labs.
Thing4 - Notice that the distance between 0 and 1 on the vertical axis is much smaller than the distance from 1 to 2, etc. To fix this we can use the *view* option in the *display* command. The *view* option tells Maple the range of the horizontal and vertical axes that you'd like to see in the plot. See below, where everything is the same as before except the very last command.

```
> sol1 := dsolve([diff(y(t),t) = -0.1·y(t)·(y(t) - 8), y(0) = 1], numeric);
  sol2 := dsolve([diff(y(t),t) = -0.3·y(t)·(y(t) - 8), y(0) = 1], numeric);
  sol3 := dsolve([diff(y(t),t) = -0.5·y(t)·(y(t) - 8), y(0) = 1], numeric);
  sol4 := dsolve([diff(y(t),t) = -0.7·y(t)·(y(t) - 8), y(0) = 1], numeric);
  p1 := plots[odeplot](sol1, 0..4, color = red); p2 := plots[odeplot](sol2, 0..4, color
      = green);
  p3 := plots[odeplot](sol3, 0..4, color = brown);
  p4 := plots[odeplot](sol4, 0..4, color = blue); display([p1, p2, p3, p4], view = [0..4, 0..8]);
```

```
sol1 := proc(x_rkf45) ... end proc
sol2 := proc(x_rkf45) ... end proc
sol3 := proc(x_rkf45) ... end proc
sol4 := proc(x_rkf45) ... end proc
p1 := PLOT(...)
p2 := PLOT(...)
p3 := PLOT(...)
p4 := PLOT(...)
```



Now label these curves with their corresponding value of k, superimpose gridlines, and give the above plot the title "My First Sweep Over a Parameter". First, copy and paste the above graph into a text box, then make the changes.



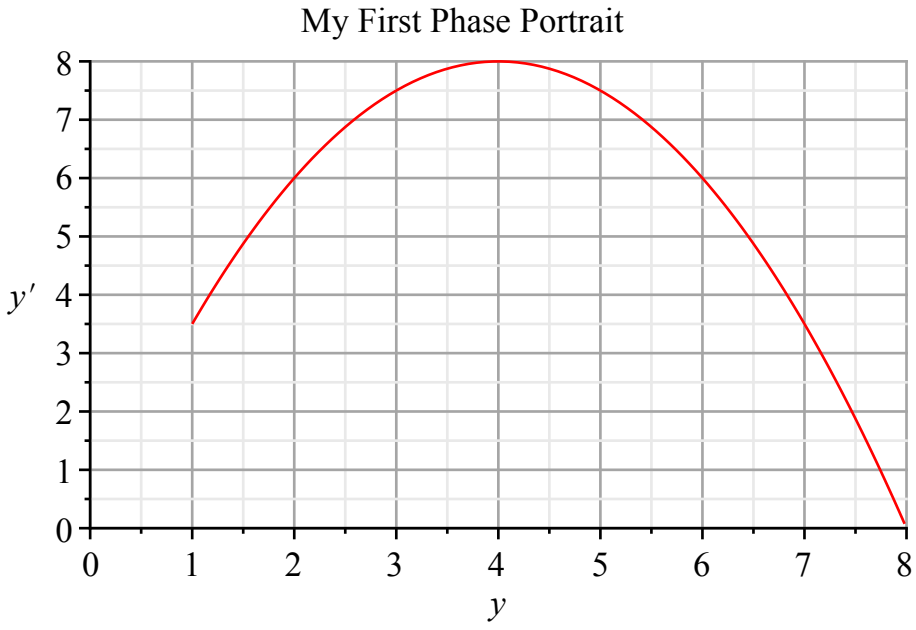
To insert labels on the solution curves, click on the above plot, select Drawing from the menu bar, and then click on the text icon **T**, which is to the right of the eraser icon. If you then click on the graph, a text box will be insert at the point you clicked. You can then start typing. Notice that you can italicize what you type or make it bold face, as you see fit. You can also type text or math into the text box by selecting the appropriate mode in the menu bar. Go ahead and label the solution curves
k=0.1, ... k=0.7. N.B. Once you have one curve labeled, you can copy-and-paste the text box and then modify it appropriately. This is a little easier than creating four new text boxes.

6. The trace feature on Plots: Suppose we wish to get an accurate approximation for the value of $y(2)$ using the above graph and that we wanted to know this value when $k=0.1$ and when $k=0.3$. Go to the above plot and click on it. Then select the third trace option (second from bottom) and put the cursor over the solution curve corresponding to $k=0.1$ and move it toward $t = 2$. You should see a big cross-hair as you do with and the coordinates of each point on the curve should be displayed as you move the cursor along the curve. Doing this we find that when $k = 0.1$, $y(2) \approx 3.3048$ and when $k = 0.3$, $y(2) \approx 7.5368$.

7. The last thing you need to know, before starting on the lab report below, is how to plot a *phase portrait*. Namely, how to get a plot of y' *vs* y. Here's how you do it.

```
> with(plots) :
> sol1 := dsolve([diff(y(t), t) =-0.5·y(t)·(y(t) — 8), y(0) = 1], numeric);
  plots[odeplot](sol1, [y(t), -0.5·y(t)·(y(t) — 8)], 0..2, color = red, view = [0 .. 8, 0 .. 8]);
```

$$sol1 := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$



Notice that in the second command above, there is a piece highlighted in yellow. This is the expression for y' because it is the right-hand side of the ODE. So, the way we tell Maple to plot y vs y' is to put them both in brackets in the plot command, but instead of writing y', we insert the appropriate expression for y'.

Based on the above graph, where does the maximum rate of change occur? I mean, for what value of y does y' achieve a maximum value? What is the maximum rate of change? Insert some gridlines to get a good approximation. Also, label the vertical axis as y' and put the title "***My First Phase Portrait***" on the Plot.

Lab 1 Report

Name: *Salt E. Dog*
Lab Partner(s): *Peter Poseidon and Nancy Neptune*
Lab Number and Name: *Lab1: Harvesting a Biomass*

```
> restart
> with(DEtools) :
> with(plots) :
> interface(typesetting = extended) :
> Typesetting[Settings](useprime, prime = t)
```

Main Body of Lab Report

In this lab you will explore different methods of harvesting a sardine biomass in the Pacific Ocean. This lab is adapted from Explorations 1.1 and 1.2 in the *ODE Architect Companion.* Here is the background for the lab.

In the two decades from 1932 to 1951, the Pacific sardine fishery completely collapsed. In this lab we will construct a mathematical model for the Pacific sardine population and use Maple to obtain solutions and answer questions about the sardine population and harvesting. As we look at different harvesting scenarios, we will consider their affect on the sardine population and try to determine the *optimal harvesting rate*, that is, the rate of harvesting that maximizes the harvest of sardines, but does not drive the sardine population to extinction.

The Pacific sardine has historically experienced long-range cycles of abundance and depletion off the West Coast of California. It was during one of the abundant periods, 1920 - 1951, that a huge sardine fishing and canning industry developed. The total catch for the California coastline reached a peak of 726,124 tons during the 1936-37 season (June through the following May). The sardine population then began a serious decline during the 1940s until, as one estimate has it, by 1959 the sardine biomass was 5% (0.2 million tons) of the 1934 level (4 million tons). (The *biomass* is the amount of a particular organism in its habitat.) There is general agreement that heavy harvesting (read *over-harvesting*) played a role in the decimation of the Pacific sardine population.

After 50 years of fishing for the Pacific sardines, a moratorium was imposed by the California legislature in 1967. The Pacific sardine seems to be making a comeback as of the mid-1980s, though the numbers are not yet near the abundant levels of the 1930s.

Given this information we now state the variables and parameters that will be included in the model, along with their units.
1. *Sardine biomass* (in units of millions of tons)
2. *Growth rate* of Pacific sardines (in units of million tons/year)
3. *Carrying capacity*, or the maximum biomass the habitat can support (in units of million tons)
4. *Sardine harvesting* (in units of million tons/year)

We are now ready to build the mathematical model. We will begin with the model that has no harvesting term and make our way towards a model that incorporates harvesting.

Exploration 1.1: Constant Harvesting

1. (No Harvesting) Let $S(t)$ represent the sardine biomass, measured in millions of tons. So, when $S = 2$ there are 2 million tons of sardines. Also, the independent variable is $t = time$, and it is measured in years. Then, the ODE that models the sardine biomass (with no harvesting) is the logistic equation given by

$S' = \frac{r S\,(6 - S)}{6}$, where we will use $r = 0.20$.

Author this ODE in a command line and give it the name *ode1*. Also, assign the above value to the parameter *r*. Then use the *Typesetting*[*Suppress*] command to suppress the independent variable *t* in the Maple output.

>

>

Use the *DEplot* command to graph sardines vs time with *t* varying from 0 to 10, *S* varying from 0 to 6, and with initial condition $S(0) = 1$. Ensure that the axes are labeled appropriately and title the graph "Sardine Biomass (no harvesting)". Use *arrows = medium, color=brown,* and *linecolor = blue* in the *DEplot* command.

>

>

Now create a graph of S' vs S. This is called a *phase portrait.* To do this, use the *dsolve* command and the *plots[odeplot]* command as we did in the Discussion section. Let *t* vary from 0..20 and use the initial condition $S(0) = 1$. You must determine what an appropriate range is for the horizontal and vertical axes and ensure that the axes are labeled appropriately. Give the plot the title "Rate of Growth versus Biomass" and put gridlines on this plot.

>

>

Once you have the graph of S' vs S, use it to verify that the rate of growth of the sardine biomass stays within the range 10% - 40% per year, depending on the size of the biomass. Briefly explain your answer. Also, what is the maximum rate of growth of the sardine biomass and what is the size of the biomass when it occurs? (include units in your answer) Again, use the phase portrait to answer this question and briefly explain.

2. (Constant Harvesting) We now modify the ODE to include a harvesting term, *h*. The new model for the sardine biomass is given by $S' = \frac{r S\,(6 - S)}{6} - h$. Here, *h* is a constant. We will take $h := 0.26$ and keep $r := 0.20$ to start with. Also, we will start with an initial biomass of $S(0) = 2.5$.

Notice that we've chosen a harvesting rate slightly less than the initial biomass growth rate. Namely, we chose *h* to be slightly less than S' when *S* is 2.5 (million tons), which was the 1941 sardine population off the coast of California. You can see from the phase portrait in part 1 above that S' is around 0.29 when *S* is close to 2.5, so this is a reasonable number to put in for the harvesting term. Before plotting, what do you expect the graph of the sardine biomass to look like and why?

After you've answered the question, use the *DEplot* command with t running from 0..10 and using the initial condition, $S(0) = 2.5$, to plot a solution curve. Choose an appropriate range for the vertical axis and title the graph "Sardine Biomass (with harvesting)". Also, once again choose *arrows=medium* and pick a *color* for the arrows and *linecolor* for the solution curve.

>

>

Describe the graph. Does it make sense? Briefly explain.

Next, we will consider how various harvesting rates impact the sardine biomass. Use the *dsolve* command with $S(0) = 2.5$ followed by appropriate plot commands (see Discussion section) to obtain a single plot with four solution curves, all starting from $S(0) = 2.5$, for harvesting levels $h := 0.1, 0.3, 0.5,$ and 0.7. If any of the solution curves become negative, adjust the range of the vertical axis so that only non-negative values of sardine biomass are plotted. Also, plot on the interval $t = 0..10$ and be sure to label each curve as its harvesting level. Do this by clicking on the plot and following the directions in the Discussion section. Finally, title the graph "Sardine Biomass (various harvesting levels)".

>

>

Once you have a plot, describe the behavior of the sardine biomass for each of these harvesting levels. Your description should be in the context of harvesting sardines, so instead of "the solution is increasing" you should say "the sardine biomass is growing", etc.

Do these solution curves make sense? Why/why not?

The "optimal harvesting level" is the largest value of h that does not jeopardize the long-term viability of the sardine population (= does NOT drive it to extinction). Based on your graph, what is the optimal harvesting level? How can the phase portrait, in problem 1 above, be used to arrive at the same conclusion?

3. How does the IC affect the optimal harvesting level. Put another way, does the optimal harvesting rate depend on the initial sardine population? To explore this, use the *DEplot* command to plot the sardine biomass for various initial conditions from 1 to 6 in increments of 0.5. You will need to insert the points [[0,1],[0,1.5],[0,2] ... [0,5.5],[0,6]] into the *DEplot* command (as we did in the Discussion section). In the ODE, set the harvesting rate, h, to the optimal harvesting level found above. Let t vary from 0..50 here and label the plot "Sardine Biomass with Optimal Harvesting)". In the *DEplot* command, set *arrows=none* and choose your favorite *linecolor* for the solution curves.

>

>

Based on your plot, is the optimal harvesting level found in problem 2 affected by the initial sardine

biomass? Explain.

Based on your graph, make a conjecture about the relationship between the optimal harvesting rate and the initial sardine biomass.

Test your conjecture by taking the initial sardine biomass to be $S(0) = 1.5$ and use harvesting rates of $h = 0.15, 0.2, 0.25$, and 0.3. Use the same set of commands as you did in problem 2 above (modified appropriately) to first solve and then plot the sardine biomass for these four harvesting levels. You should end up with one plot that includes all four solution curves. Be sure to label the solution curves with their corresponding harvesting level and put a title on the graph. Let t vary from 0..10 here.

>

>

Does this graph confirm your conjecture? What is an approximate value of the optimal harvesting rate for the initial population $S(0) = 1.5$? Briefly explain how you got it.

>

>

Exploration 1.2: Constant Effort Harvesting

1. We now change the model so that the harvesting term represents landing a certain percentage of the existing sardine biomass each year. This is known as *constant effort harvesting*, and is incorporated into the model by setting the harvesting term, $h = 0.25 \cdot S(t)$. Namely, we will harvest 25% of the sardine biomass each year.

Author the appropriate ODE (with this new harvesting term) on a command line and use the *DEplot* command to obtain solutions for several initial values of sardines, say, $S = 2, 4, 6,$ and 8. Let t vary from 0..30 and set *arrows=medium*. Choose your favorite colors for arrows and solution curves and put a title on your graph.

>

>

Based on your graph, describe what happens to the sardine biomass for different initial conditions.

We will now explore three harvesting levels, namely, 0.15*S, 0.10*S, and 0.05*S in an attempt to determine which harvesting level is the best one to use. Namely, which one will allow for the maximum number of sardines to be harvested without driving the sardine population to extinction.

Using a set of commands similar to what you used above (but appropriately modified), solve the

ODE with the three harvesting terms above and graph all three solutions on one set of axes. Take the initial sardine biomass to be 8 each time you solve the ODE and let time go out to $t = 100$. Superimpose grid lines on the graph and give it a title. Also, be sure to annotate your graph to make it clear which solution curve corresponds with which harvesting level.

>

>

Which harvesting level maximizes the number of sardines harvested? In other words, which harvesting level is the "optimal harvesting level"? Briefly explain. In your explanation, you should state what the *sardine biomass* is at $t = 100$ for each of the three harvesting levels. Give approximate number based on your graph. Also, state the amount of *sardines harvested* at $t = 100$ for each of the three harvesting levels. (You will have to do the computation.) The easy way to do this is to use your graph to complete the table below. (Include units in your answer.)

Harvesting Level	**Sardine Biomass**	**Sardines Harvested**
.05*S(t)		
.10*S(t)		
.15*S(t)		

2. (How does the IC affect the optimal harvest percentage?) To determine whether the optimal harvesting percentage depends on the IC, solve the constant effort harvesting model for different initial populations. For the first run, set the initial population to $S(0) = 6$ and take the harvesting term to be 0.15*S, 0.10*S, and 0.05*S. As before, plot all these on the same set of axes, title the graph, label the harvesting levels, and let t vary from 0..100.

>

>

Based on your graph, what is the optimal harvesting percentage for the initial condition $S(0) = 6$? Include a brief explanation and possibly a table.

Now repeat this process with the same harvesting levels as before and for the initial condition $S(0) = 4$. What is the optimal harvesting percentage for this IC?

>

>

How does this result (for constant effort harvesting) compare with your result in problem 3 (for constant harvesting)?

Conclusions:

Catch-all:

Time Required:

Appendix II: Lab 2 - Slope Fields and Skydiving

Discussion for Lab 2

The following commands will be useful to you in completing Lab 2. Some of the commands we will go over in the discussion section are being revisited from Lab 1 and others are new. All will help you at one point or another in Lab 2.

We will start with a few reminders. The first reminder is that you will start every lab with the following commands. Go ahead and execute them now.

```
> restart
> with(DEtools) :
> with(plots) :
> interface(typesetting = extended) :
> Typesetting[Settings](useprime, prime = t)
```

$$true, x \tag{1.1}$$

```
>
```

The second reminder is about how to author and name an ODE. This is, in some sense, the most basic thing you'll be doing in Maple. Here's an example.

```
> ode1 := diff(y(t), t) = y(t)·(3·y(t) − 60)
```

$$ode1 := y'(t) = y(t)\ (3\,y(t) - 60) \tag{1.2}$$

To suppress t from being displayed as the argument for y and y' use the following command:

```
> Typesetting[Suppress](y(t))
```

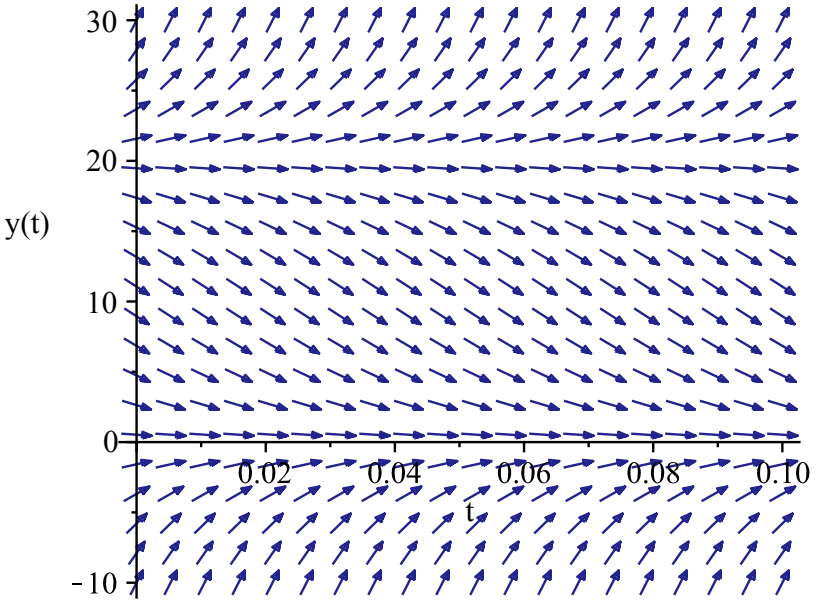
Now check.

```
> ode1
```

$$y' = y\ (3\,y - 60) \tag{1.3}$$

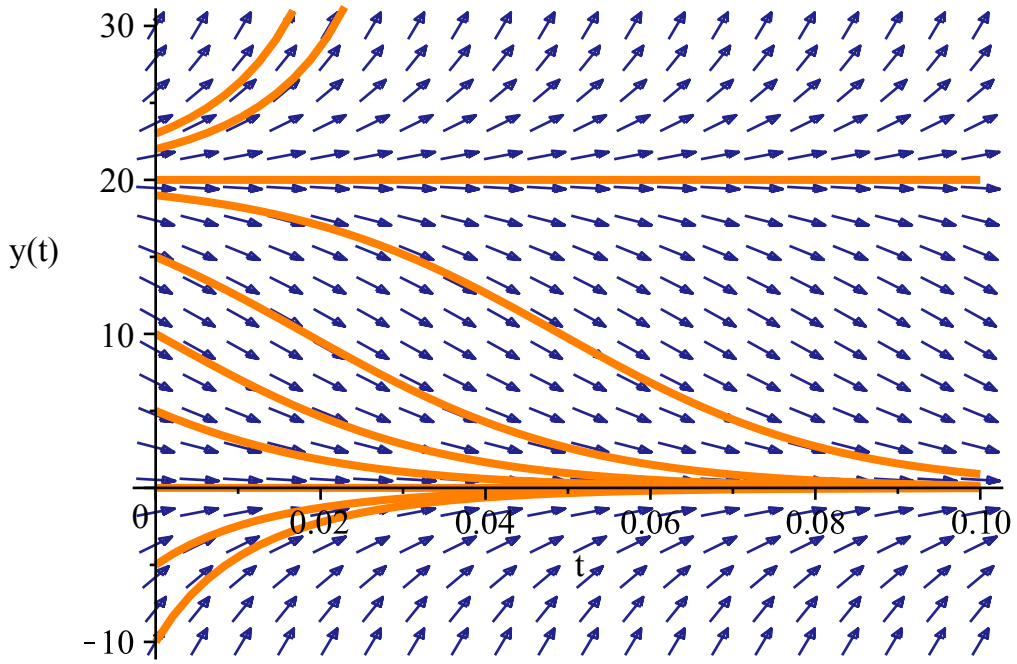
To plot a direction field you use the DEplot command. Here's an example.

```
> DEplot(ode1, y(t), t = 0..0.1, y = -10..30, arrows = medium, color = navy)
```



Plotting solution curves helps us to see the long-term behavior of the solutions. You have to decide which ICs are helpful and then put those ICs into the *DEplot* command. From the above direction field, it looks like solutions will move toward $y = 0$ and away from $y = 20$ so I will choose some ICs to confirm or refute this. See below command line.

```
> DEplot(ode1, y(t), t = 0..0.1, y = -10..30, [[0,-10], [0,-5], [0, 0], [0, 5], [0, 10], [0, 15],
  [0, 19], [0, 20], [0, 22], [0, 23]], arrows = medium, color = navy, linecolor = coral)
```



Notice the time scale in the above example. Things are happening pretty quickly. The solution curves converge to $y = 0$ even on the very short interval $t = 0..0.1$! Moral of the story: It is NOT always obvious what the appropriate time interval is. Sometimes you don't want to go past 0.1 or 0.2 and other times you have to go out to 100. You just have to play around a bit.

You will need to plot solution curves for different parameter values in this lab. Recall from Lab 1, that when you want to keep the IC fixed and vary a parameter, you use the *dsolve* and *plots [odeplot]* set of commands. You must type the specific parameter value into each *dsolve* command, get the right range for the time variable in the *plots[odeplot]* command, make sure the IC is the same in all the *dsolve* commands, and use the *view* option in the *display* command, as needed.

Here is an example involving the ODE for the falling body problem with linear drag where I've let the drag coefficient, k, vary from 0.5, 1, ... 2.5 and I took the initial velocity to be -25 ft/sec, namely, the skydiver jumps vertically upward with a speed of 25 ft/sec as he exits the plane.

```
> sol1 := dsolve([diff(v(t), t) = 32 - (0.5/5)*v(t), v(0) = -25], numeric); sol2
       := dsolve([diff(v(t), t) = 32 - (1/5)*v(t), v(0) = -25], numeric); sol3
       := dsolve([diff(v(t), t) = 32 - (1.5/5)*v(t), v(0) = -25], numeric); sol4
       := dsolve([diff(v(t), t) = 32 - (2/5)*v(t), v(0) = -25], numeric); sol5
```

$$:= dsolve\left(\left[diff(v(t),t) = 32 - \left(\frac{2.5}{5}\right)\cdot v(t), v(0) = -25\right], numeric\right); p1$$

$:= plots[odeplot](sol1, 0..30, color = red); p2 := plots[odeplot](sol2, 0..30, color = green); p3 := plots[odeplot](sol3, 0..30, color = brown); p4 := plots[odeplot](sol4, 0..30, color = blue); p5 := plots[odeplot](sol5, 0..30, color = coral); display([p1, p2, p3, p4, p5], view = [0..30, -25..300]);$

$sol1 := \mathbf{proc}(x\_rkf45) \ldots \mathbf{end\ proc}$

$sol2 := \mathbf{proc}(x\_rkf45) \ldots \mathbf{end\ proc}$

$sol3 := \mathbf{proc}(x\_rkf45) \ldots \mathbf{end\ proc}$

$sol4 := \mathbf{proc}(x\_rkf45) \ldots \mathbf{end\ proc}$

$sol5 := \mathbf{proc}(x\_rkf45) \ldots \mathbf{end\ proc}$

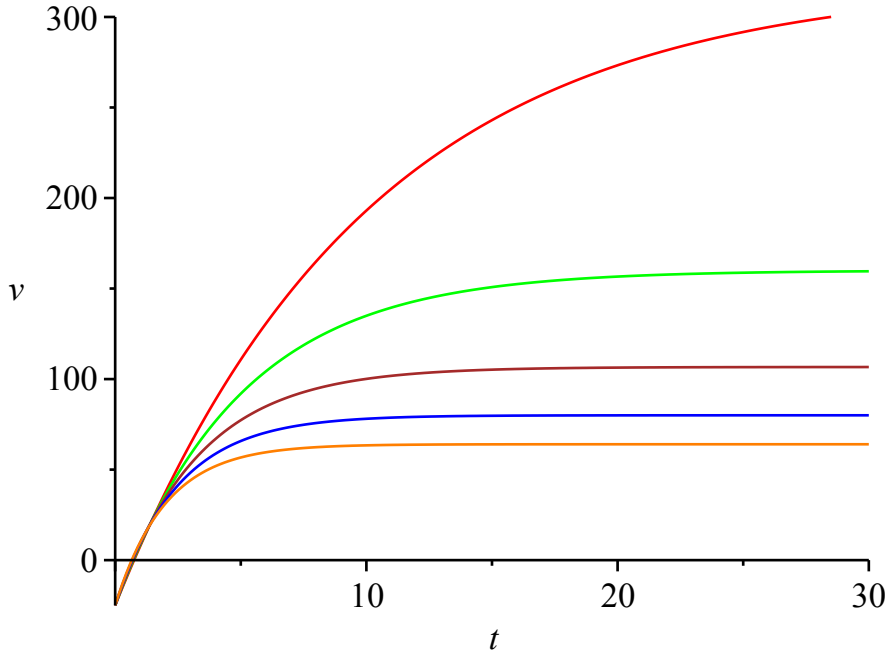
$p1 := PLOT(\ldots)$

$p2 := PLOT(\ldots)$

$p3 := PLOT(\ldots)$

$p4 := PLOT(\ldots)$

$p5 := PLOT(\ldots)$



Every now and then it's helpful to *unassign* values that you've previously assigned to a parameter. To do this you use, oddly enough, the *unassign* command. Here's an example.

> $Q := a \cdot x^2 + b \cdot x + c$

$$Q := a\,x^2 + b\,x + c \tag{1.4}$$

> $a := 2; b := -3; c := 9$

$$a := 2$$
$$b := -3$$
$$c := 9 \tag{1.5}$$

> Q

$$2\,x^2 - 3\,x + 9 \tag{1.6}$$

```
> unassign('a'); unassign('c')
> Q
```

$$a\,x^2 - 3\,x + c \tag{1.7}$$

When dealing with exponential functions, you will need to author an expression with e in it. You can NOT simply type *e* from the keyboard because Maple will treat it as the variable *e*, not the constant e ≈ 2.718281828. To get the right e, you can either click on the e in the Common Symbols palette (to bring it into your Maple worksheet) or you can type *exp(-t)* in the command line. Here's an example involving e^{-2t}.

```
> w := t→e^(-2·t)
```

$$w := t \mapsto e^{-2\,t} \tag{1.8}$$

```
> z := exp(-2·t)
```

$$z := e^{-2\,t} \tag{1.9}$$

```
> w(1)
```

$$e^{-2} \tag{1.10}$$

```
> z(1)
```

$$\left(e^{-2\,t}\right)(1) \tag{1.11}$$

```
> subs(t = 1, exp(-2·t))
```

$$e^{-2} \tag{1.12}$$

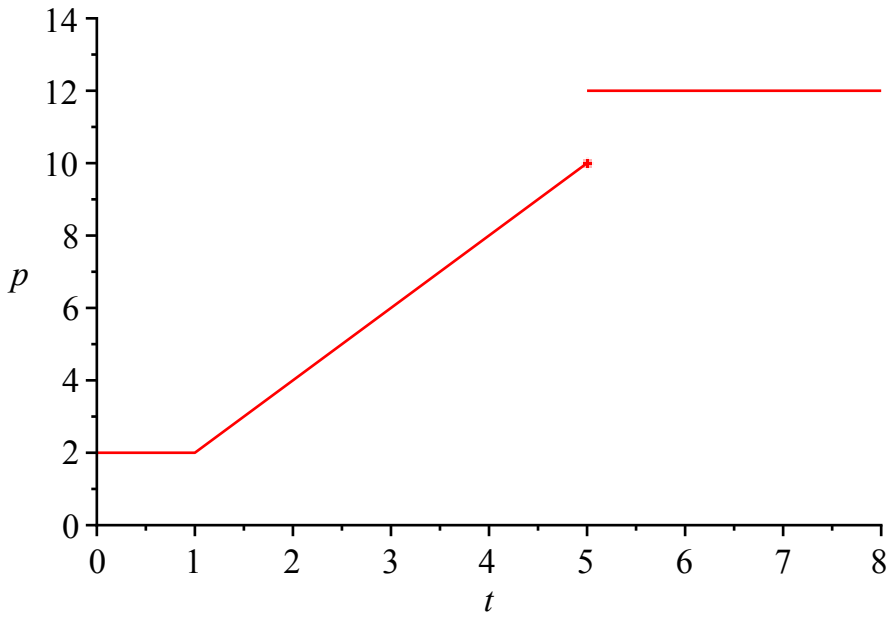
Recall that *w* above is a *function*, whereas *z* is an *expression*. In Maple, we evaluate functions (in the natural way) and we use the *subs* command to substitute values into expressions.

Now for something new. Towards the end of the lab, you will need to solve an ODE involving a piecewise defined function.
The *piecewise* command take the form *piecewise(condition1, f1,condition2,f2,condition3,f3),* where *condition1* is a relation or a boolean combination of inequalities and *f1* is some function (or constant) and likewise for the other conditions and functions. Although I stopped at *condition3,f3* in the above command, you can string as many of these together as you like. Here's an example where I first author and then plot a piecewise function.

```
> p := t → piecewise(t < 1, 2, t ≥ 1 and t ≤ 5, 2 t, t > 5, 12)
```

$$p := t \mapsto \begin{cases} 2 & t < 1 \\ 2\,t & 1 \le t \le 5 \\ 12 & 5 < t \end{cases} \tag{1.13}$$

```
> plot(p(t), t = 0..8, p = 0..14, discont = true)
```



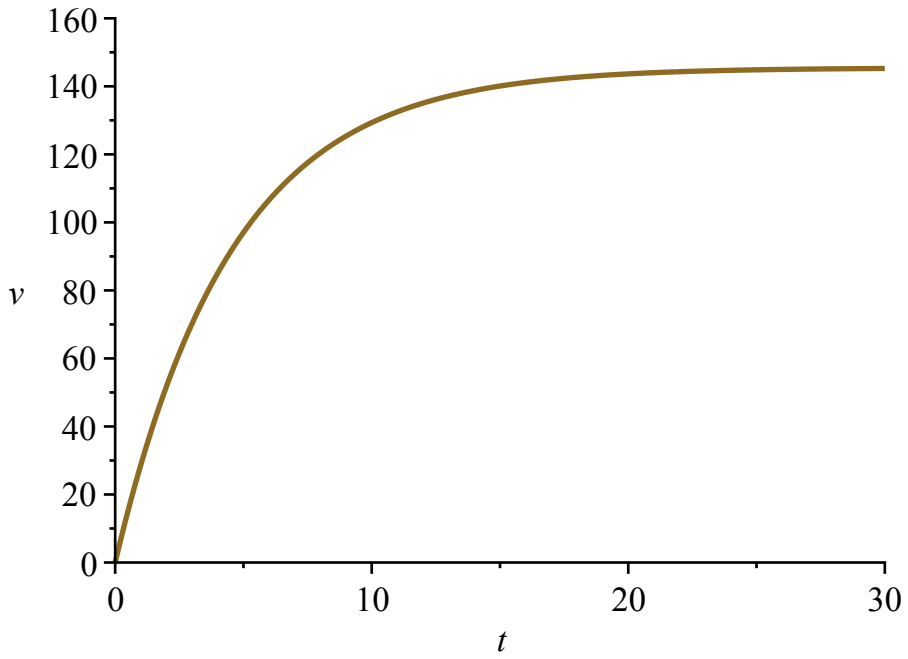
The *dsolve* and *plots[odeplot]* commands gives you a lot of flexibility with respect to plotting. What follows is three examples to illustrate this. We'll start with using the *dsolve* command to plot velocity as a funtion of time. Keep an eye on the piece highlight in orange. It will be of the general form $[x(t), y(t)]$, where $x(t)$ is the variable to be plotted on the x-axis and $y(t)$ is the variable to be plotted on the y-axis.

```
> k := 1.1
```

$$k := 1.1 \tag{1.14}$$

```
> sol1 := dsolve([diff(v(t), t) = 32 - (k/5)·v(t), v(0) = 0], numeric);
  plots[odeplot](sol1, [t, v(t)], 0..30, color = red, view = [0 .. 30, 0..160], thickness = 2,
      numpoints = 1000, color = sienna);
```

$$sol1 := \mathbf{proc}(x\_rkf45) \ \ldots \ \mathbf{end\ proc}$$



Now we'll use the *dsolve* and *plots[odeplot]* commands to plot the acceleration as a funtion of time.

> $k := \frac{13}{10}$

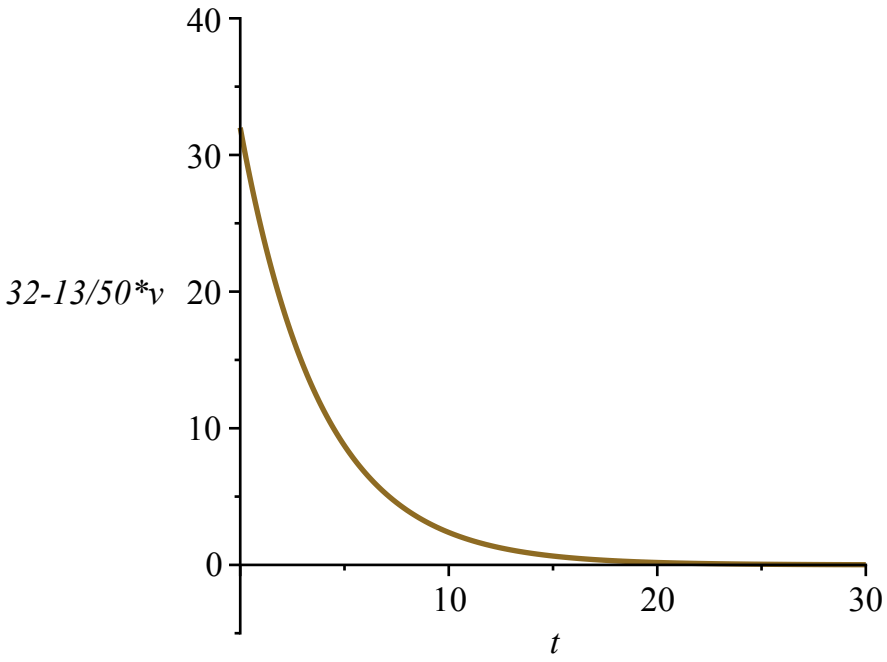
$$k := \frac{13}{10} \tag{1.15}$$

> $sol1 := dsolve\left(\left[diff(v(t), t) = 32 - \left(\frac{k}{5}\right)\cdot v(t), v(0) = 0\right], numeric\right);$

$plots[odeplot]\left(sol1, \left[t, 32 - \left(\frac{k}{5}\right)\cdot v(t)\right], 0..30, color = red, view = [0 .. 30, -5..40],\right.$

$\left.thickness = 2, numpoints = 1000, color = sienna\right);$

$sol1 := \mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc}$



Notice that the second command above tells Maple to plot t vs $a(t)$ by using brackets, namely, [t, $a(t)$], but instead of typing $a(t)$, we type in the expression for $\frac{dv}{dt}$, which is the R.H.S. of the ODE $v' = 32 - \frac{k\,v}{5}$.

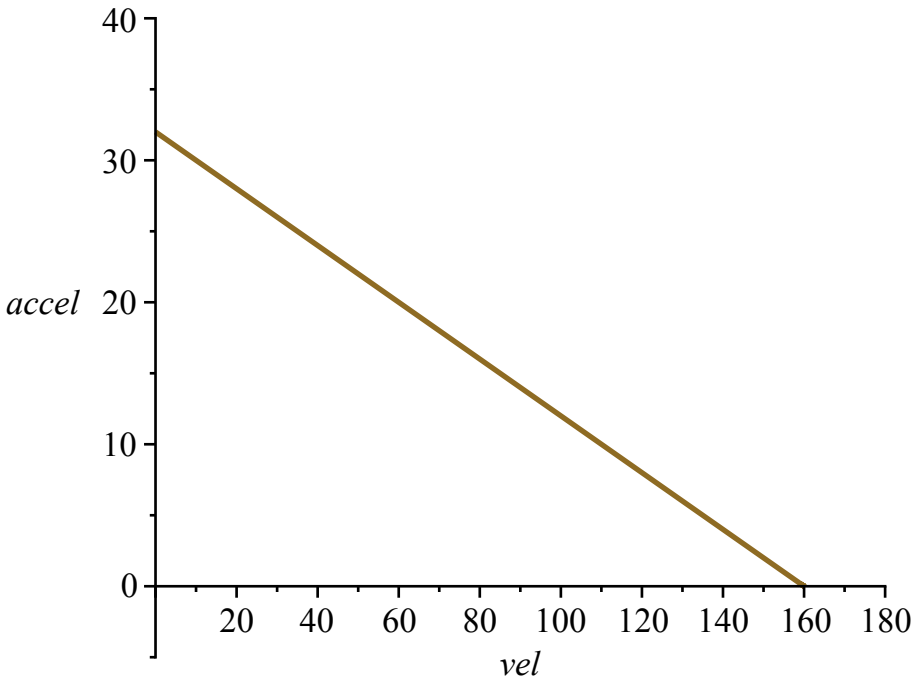
Finally, we'll use the *dsolve* command to plot a phase portrait of the velocity, namely, to plot v vs v' or velocity vs acceleration.

```
> k := 1.0
```

$$k := 1.0 \qquad \textbf{(1.16)}$$

```
> sol1 := dsolve([diff(v(t),t) = 32 - (k/5)*v(t), v(0) = 0], numeric);
  plots[odeplot](sol1, [v(t), 32 - (k/5)*v(t)], 0..50, color = red, view = [0 .. 180, -5..40],
      labels = [vel, accel], thickness = 2, numpoints = 1000, color = sienna);
```

$$sol1 := \textbf{proc}(x\_rkf45) \ \ldots \ \textbf{end proc}$$



You are now ready to start Lab 2. So, save this file to your H: drive as Lab 2 Discussion, then save it again as Lab 2 Report, then delete the Discussion piece from your Lab 2 Report file, and finally, open the Lab 2 Discussion file from the H: drive (so you have access to the Discussion section as you do your lab).

Remember to shrink your graphs and save your Maple file frequently as you go.

Lab 2 Report

Header Information
Name: *Cat in the Hat*

Lab Partner(s): *Thing 1 and Thing 2*

Lab Number and Name: *Lab2: Slope Fields and Skydiving*

```
> restart
> with(DEtools) :
> with(plots) :
> interface(typesetting = extended) :
> Typesetting[Settings](useprime, prime = t)
```

Main Body of Lab Report

In this lab you will explore the behavior of the solutions of several ODEs using Maple to plot slope fields and solution curves. You will also work through a skydiving problem. This lab is adapted from Explorations 2.2 and 2.4 in the *ODE Architect Companion.*

Exploration 2.2: Direction Fields

1. *What happens in the long term?*

For each of the ODEs below, describe how the solution behaves "as t gets very large". Namely, describe the *asymptotic* behavior of the solution. Include plots of direction fields only and direction fields with solution curves, as directed. Do this in three steps.

Step 1: Author the ODE and plot the direction field. Based on the direction field, make a conjecture about the asymptotic behavior of the solutions.

Step 2: Based on the direction field, pick 6 - 8 ICs that would help you confirm or refute your conjecture and then plot the direction field and solution curves for these ICs. Once you've done this, either update your conjecture or state why the graph supports it.

Step 3: State whether the asymptotic behavior of the solutions depends on the ICs and if so, how so.

(a) The first ODE we will work with is $y' = y - 20$. Author this ODE and name it *ode1*. Use the *Typesetting[Suppress]* command to suppress the t dependence in the Maple output. Then plot the direction field on $t = 0..2$ and $y = -100..100$.

```
>
>
```

(b) The next ODE we will work with is $y' = \sin\left(\frac{\pi \cdot y}{10}\right)$. Author this ODE and call it *ode2*. Graph the direction field for $t = 0..15$ and $y = -30..30$.

```
>
>
```

(c) The next ODE we'll work with is $y' = \frac{(y - t)}{10}$. Author this ODE and call it *ode3*. Then graph the direction field for $t = 0..25$ and $y = -60..60$.

```
>
>
```

2. *Strange solutions.*

Make up your own ODEs, especially ones whose solution curves or slope fields form strange

patterns. Use Maple to display your results. Be creative here!

More specifically, you should make up two different ODEs and include the direction field for each with 6 - 8 solution curves carefully chosen to illustrate the behavior of the solutions for various ICs. In each case, describe the behavior of the solutions (in general) and the asymptotic behavior of the solutions (in particular). Also, state whether the asymptotic behavior depends on the ICs and if so, how so. You will need to choose an appropriate range for t and y for your plots. Keep in mind that if your graph is "jagged" or "choppy", you can make it "smooth" by using *numpoints =500* or *numpoints = 1000* in your plot command.

>

>

Exploration 2.4: The Skydiver

1. *Terminal speed of a falling body.*
In this exploration you will use Maple to determine a skydiver's terminal velocity for several different values of the viscous damping coefficient (use m = 5 slugs and g = 32 ft/sec^2). The ODE that will serve as our model for this falling body problem is $v' = 32 - \left(\frac{k}{5}\right)\cdot v$.

Keep in mind that for falling body problems, the downward direction is taken to be the $+x$ direction . This means that x increases as the body falls. It also means that the velocity, $v(t)$ increases in the downward direction, as does the acceleration, $a(t)$.

To get started, author the above ODE and solve it for different values of the parameter k, say, $k = 1$, 2, 3, 4, and 5. You are trying to determine whether the terminal velociy depends on the parameter k. To do this, you'll need to use the *dsolve* command and the *plots[odeplot]* command. In the *dsolve* commands take the initial condition to be $v(0) = 0$ (the body starts from rest) and in the *plot [odeplot]* commands, let t vary from 0..20. Once you have a plot, label the solution curves with their corresponding value of k and put gridlines on the plot.

Based on your graph, does the terminal velocity depend on k? Briefly explain.

>

>

Is there any difference (in terminal velocity) if the skydiver jumps from 25,000 ft instead of 13,500 ft? To answer this we must get the height, $h(t)$, into the problem by noting that the ODE that models the height above ground is given by $h' = -\frac{160}{k} + \left(\frac{160}{k}\right)\cdot\exp\left(-\frac{k\cdot t}{5}\right)$. Author the appropriate ODE for $h(t)$ and solve it using the *DEplot* command to obtain a graph of $h(t)$ vs t. Your should use the ICs $h(0) = 13500$ and $h(0) = 25000$ for your graphs. Also, take $k = 1$ here and let time run from $t = 0..50$.

Based on these graphs, does the terminal velocity change when the skydiver jumps from different heights? Hint: It may be helpful to note that $h(t) = x(0) - x(t)$, which means that

$\frac{dh}{dt} = -\frac{dx}{dt} = -v(t)$. Keep this in mind as you look at the graph of $h(t)$ to determine the long-term behavior of the velocity for different initial heights.

2. *Slow down!*

If a skydiver can survive a free-fall jump only if she hits the ground at no more than 30 ft/sec, what values of k make this possible? Are these k-values realistic?

We are now back to using the ODE $v' = g - \left(\frac{k}{m}\right) \cdot v$, with m=5 and g=32. Author this ODE and use the *dsolve* command to solve it for several values of k. For each solution, take the IC to be $v(0) = 0$. Consider values of k between 2 and 6, say $k = 2, 3, 4, 5,$ and 6. Let time vary from 0..20 and put gridlines on your plot. Also, label the solution curves with the corresponding value of k.

Based on your graph, what value of k allows the sky diver to survive? Is it realistic?

3. *A Modeling Challenge!*

In this part, we construct a model of a parachute that opens over a 3 second period of time. Hence, the viscous damping coefficient is a function of time, $k(t)$. We assume that the skydiver jumps from an altitude of 13,500 ft and the parachute opens after 65 seconds of free-fall. We also assume that the damping coefficient changes "linearly" from $k = 0.86$ before the parachute opens to k =6.71 after the parachute deploys. So, the governing ODE here is $v' = 32 - \left(\frac{k(t)}{5}\right) \cdot v$.

The trick here is to author the right expression for the damping coefficient, $k(t)$. Start by authoring the appropriate function for $k(t)$ and then graph $k(t)$ on the interval $t = 0..100$ to make sure it fits the above description. Hint: The easiest way to do this on Maple is to use the *piecewise* command to define k as a piecewise function of t. See the example in the Discussion section and adapt it to this problem.

Go ahead and author the ODE for $v(t)$ with the appropriate term for $k(t)$ and use the *dsolve* and *plots [odeplot]* commands to solve the ODE and graph the solution. Let t run from 0..100 and put gridlines on your plot. Also, take the IC to be $v(0) = 0$. Once you have the graph of $v(t)$ vs t describe (in words) what is happening to the skydiver's velocity before, during, and after the parachute opens. Does your graph make sense physically?

Now modify the commands you just used to plot the velocity in order to obtain a graph for the acceleration, $a(t)$, of the skydiver You only need to modify one piece of the *plots* command. Here's a hint: the acceleration, $a(t) = \frac{dv}{dt}$, which is the right hand side of the ODE we've been working with. (Hence, you'll need to use the same set of commands as you did to plot the phase portrait in Lab 1, appropriately modified of course.) Let t run from 0..100 and put gridlines on your plot. Also, take the IC to be $v(0) = 0$. Once you have the graph of $a(t)$ vs t describe (in words) what is happening to the sky diver's acceleration before, during, and after the parachute opens. Does your graph make sense physically?

>

>

Conclusions:

Catch-all:

Time Required: