MATHEMATICA MANUAL

FUNDAMENTALS OF DIFFERENTIAL EQUATIONS
EIGHTH EDITION

FUNDAMENTALS OF DIFFERENTIAL EQUATIONS AND BOUNDARY VALUE PROBLEMS
SIXTH EDITION

R. Kent Nagle

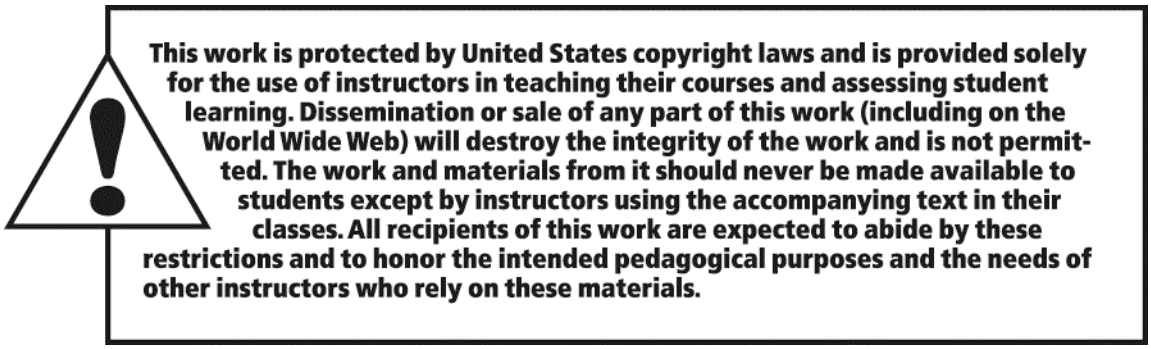
Thomas Polaski
Winthrop University

Edward B. Saff
Vanderbilt University

Arthur David Snider
University of South Florida

Addison-Wesley
is an imprint of

PEARSON



Reproduced by Pearson Addison-Wesley from electronic files supplied by the author.

ISBN-13: 978-0-321-74838-6
ISBN-10: 0-321-74838-7

Addison-Wesley
is an imprint of

PEARSON

www.pearsonhighered.com

Preface

This manual is designed to be used specifically with the book *Fundamentals of Differential Equations*, Eighth Edition, by Nagle, Saff, and Snider and the companion book *Fundamentals of Differential Equations and Boundary Value Problems*, Sixth Edition, by Nagle, Saff, and Snider.

This manual contains valuable information for incorporating computer technology into the teaching and learning of the theory and applications of ordinary differential equations. Specifically it contains information of how to use the computer algebra system *Mathematica.* It follows as closely as possible the fine *Maple* manual produced for the above text.

Included in this manual is all the information you will need to use *Mathematica* in the instruction of differential equations. (When we refer to *Mathematica* in this manual we are referring specifically to *Mathematica* 8.0. Some of the commands may not work on earlier versions of *Mathematica*, although users of version 7.0 ought not to have any problems.) The first part of the text contains a general introduction on the use of *Mathematica.* A chapter giving demonstrations on the use of *Mathematica* in calculus and differential equations follows this introduction. Then follow fourteen laboratories which comprise the heart of the manual. These labs are designed to aid in the understanding of central concepts studied in a standard introductory course in differential equations. Each lab introduces the concepts and demonstrates how *Mathematica* is used in the context, but then requires the students to become involved in exploratory exercises and questions. The objectives are clearly stated so that the instructor can choose which labs are most appropriate for the course he or she is teaching. The text closes with suggestions for additional projects for students to do, if desired. The additional projects were originally compiled by Douglas B. Meade of the University of South Carolina.

I hope that the information and ideas presented in this manual will be useful and interesting, to you and to your students in differential equations courses. Comments, corrections, and suggestions for improvement in the manual are welcome.

Thomas W. Polaski
Winthrop University
April 2011

Table of Contents

1. Introduction

The study of mathematics can be greatly enhanced with the use of technology. Differential equations, in particular, is a subject in which concepts can be more thoroughly explored with the use of technology. The reason for the importance of technology is that differential equations in many instances involve parameters that can be varied. In addition, they usually involve symbolic and numerical calculations, many of which may be intractable without the computer. Moreover they have solutions that should be visualized graphically to understand behaviors.

Among the benefits of using technology in the teaching of differential equations are the following. The student

- can see in easy fashion the behavior of solutions by means of graphs in the form of direction fields and phase planes;
- will come to understand the role that parameters play in differential equations;
- will be able to examine “what-if” situations to discover properties of differential equations;
- will be able to handle computationally intractable problems and generate numeric solutions; and
- will be able to solve various applications of differential equations that involve extensive computations.

This manual is written for the use of the computer algebra system *Mathematica* in differential equations. The version of *Mathematica* used is *Mathematica* 8.0. *Mathematica* contains built-in libraries that have many routines and commands useful in studying differential equations. Information on the use and syntax of these commands and routines is available in a rather extensive and very readable help directory.

When working with *Mathematica*, one is presented with a notebook on which there can be combined text, input, output, and graphics. Each section of this manual is in fact a *Mathematica* notebook. With ease, commands and text can be edited and corrected. It is easy to copy and paste just as in a word processor. This makes it easy for students to do the mathematics in *Mathematica* in the notebook, display the results and the graphs, and then to make comments and summaries of what they have learned.

There are two introductory chapters that detail how to use *Mathematica.* One is a general introductory chapter “Introduction to *Mathematica*” and the other is a more specific introduction “Demonstrations on Using *Mathematica* in Calculus and Differential Equations.” Both of these can be used for reference when the labs are used from the manual. However, a preliminary reading of them together with trial executions of the commands given is very beneficial for what follows in the manual.

Following the introductory chapters are the laboratories, which comprise the heart of the manual. These cover most of the essential topics in an introductory course in differential equations. Some labs are longer than others; some require more work than others. Each lab introduces the concepts and demonstrates how *Mathematica* is used in the context, but then requires the students to become involved in exploratory exercises and questions. It may be appropriate to only use part of a given lab rather than to do all the exercises. The instructor should use discretion on how to best use the labs and modify them as needed for a given situation.

The last part of the manual provides additional project ideas. The projects given represent diverse kinds of applications including chemical engineering, mathematical epidemiology,special functions, electrical circuits, and athletics. A brief overview of each project is given at the beginning. Students are also encouraged in this chapter to devise their own projects and a process for doing this is given.

In summary, the objective of this manual is to help students use technology in the form of *Mathematica* to aid them in understanding and enjoying the study of differential equations. Although some effort must be exerted to learn the basics of *Mathematica* and work is needed to accomplish the labs, the rewards are well worth the effort. Technology can greatly aid in the comprehension of mathematics.

2. Introduction to *Mathematica*

What is *Mathematica*? *Mathematica* is a computer algebra system (CAS). What is a CAS? As the name suggests, it is a computer software program that manipulates sophisticated algebraic symbols and expressions -- saving you the time, the tedium, and the frustration of doing these by hand. But as we shall see, it is much more. It can do some very advanced, difficult calculations, generate impressive graphs, and generate numerical and symbolic solutions to problems that may be intractable by hand. If used properly, a CAS allows you the user to concentrate more on the concepts of mathematics rather than spending time doing endless, time-consuming computations.

A CAS is a software package that works directly with an expression as a basic data structure (compare this to most software packages that require numeric data and perform numeric operations on the data). *Mathematica* is one of several CASs; other major examples are Maple, Derive, MATLAB, and MathCad.

In this introductory section, the rudiments of using *Mathematica* are given. Discussed in this section are the following topics.

1. The *Mathematica* Environment
2. Essentials for Using *Mathematica*
3. Functions and operators
4. Plotting with *Mathematica*
5. Using *Mathematica* Packages
6. Programming with *Mathematica*

2.1 The *Mathematica* Environment

When you use *Mathematica* on the computer, you work in a *Mathematica* notebook. This document was written in a *Mathematica* notebook. It is the means by which you input, output, save, and print information. A *Mathematica* notebook consists of a list of cells, each of a particular type: text, *Mathematica* input, *Mathematica* output, graphics or animation. The limits of each cell are denoted by square brackets which appear in the right margin of the notebook. These cell delimiters are not generally printed. Cells may be removed by left-clicking on the bracket and pressing DEL.

Computing in *Mathematica* is done by executing a cell in the notebook containing input. You execute a cell by placing the cursor within the cell, and pressing SHIFT and ENTER simutlaneously on the QWERTY keypad or by pressing ENTER alone on the numeric keypad. Unless told otherwise, executing an input cell generates a corresponding output cell. The following is an example of an input cell containing a *Mathematica* command and the output which is produced. It is an integration command.

```
Integrate[x^2, x]
```

$$\frac{x^3}{3}$$

Note that the arguments of all *Mathematica* functions are enclosed in square brackets, and that the names of built-in *Mathematica* functions begin with capital letters. *Mathematica* is a case-sensitive language; lower-case letters and upper-case letters are distinct.

There are basically three types of cells in a notebook: (1) inert text, (2) *Mathematica* input, and (3) *Mathematica* output.

1. text or inert input: The cell you are reading now is an example of text. *Mathematica* commands which occur in such cells are not executable, even if the font used looks like *Mathematica* input. For example, to find **`2*ArcSin[1]`** you must place it in an input cell. You can do this by the standard copy command CTRL-C, then moving the cursor to a space between cells. The cursor will become horizontal when you are between cells. Left-click now to produce a complete horizontal line, then use the standard paste command CTRL-V. *Mathematica* always assumes that new cells are input cells, so we now have an input cell which may be evaluated:

```
2 * ArcSin[1]
```

π

You can also enter text or input by using *palettes*. Palettes contain many specialized symbols and alternative alphabets which may be useful. For our purposes, you will probably only need the Basic Math Assistant palette. To open this palette, go to the Palettes menu at the top of the *Mathematica* window. Clicking on Basic Math Assistant will open a window at the right of your screen. Left-clicking on an entry in the palette places it in the notebook at the cursor's current position. Some palette entries like $\square^{\square}$call for your input. For example, to type x^2 ,you first click on $\square^{\square}$. The base of this expresssion should be filled. Type x to fill the base, then TAB, then 2. Hitting the right arrow key will return you to standard text entry mode.

2. *Mathematica* input**:** Input is a mathematical statement you want *Mathematica* to evaluate. The input text is in **`bold Courier New`** font. *Mathematica* evaluates the command when you hit SHIFT and ENTER simultaneously on the QWERTY keypad or by pressing ENTER alone on the numeric keypad. You can also enter input commands using palettes, so expressions like

```
α + β
```

$\alpha + \beta$

are possible.

3. *Mathematica* output**:** Output, by default, is in `Courier New` font.

The menus displayed at the top of the *Mathematica* window contain pre-programmed commands that can enhance and expedite the creation of a *Mathematica* notebook. A description of some of the options is provided. Other commands will be addressed when the need arises.

- **The File menu**

New:	Selecting "Notebook (.nb)" creates a new, untitled *Mathematica* notebook.
Open...:	Opens an existing *Mathematica* notebook.
Close:	Closes the current *Mathematica* notebook.
Save:	Saves the current *Mathematica* notebook.
Print...:	Prints the current *Mathematica* notebook.
Exit:	Exits *Mathematica*. You will prompted if you wish to save unsaved work.

In addition, a list of recently opened notebooks appears in this menu for quick re-opening.

The Edit menu

Undo:	Undoes the last typing sequence.
Cut:	Removes selected text and copies it to the Clipboard.
Copy:	Copies selected text to the Clipboard.
Paste:	Inserts the contents of the Clipboard at the cursor.
Find...:	Searches the current notebook for matches to the text contained in the "Search for:" field of the dialog box. Also allows for replacement of the selected text.

The Cell menu

Delete All Output:	Deletes all cells in the current notebook that have been produced as output.

The Evaluation menu

Abort Evaluation:	Aborts the current evaluation.

The Help menu

If you need **help**, the on-line help for *Mathematica* is very good -- once you learn how to use it. The help pages describe the syntax of each command, a brief description of the algorithm used to implement the command, and a few examples illustrating the use of the command. The examples can be very helpful. One can often evaluate a cell in a help workbook and then cut-and-paste it into the current notebook, confirm that the example works as advertised, then modify the arguments to answer the question at hand.

You can get help while in a notebook without using the Help menu. For example, to obtain help on the command **`Plot,`** type **`? Plot`** in an input cell and evaluate.

```
? Plot
```

Plot[f, {x, x_{min}, x_{max}}] generates a plot of f as a function of x from x_{min} to x_{max}.
Plot[{f_1, f_2, ...}, {x, x_{min}, x_{max}}] plots several functions f_i. ≫

Alternatively, you can go the Help menu on top of the screen and select one of these options and select either Documentation Center or "Find Selected Function." To search for a particular *Mathematica* function, type it in the dialog box. An index of functions is also available.

2.2 Essentials for using *Mathematica*

At the beginning of each *Mathematica* session, all variables have no value attached to them. It is a good idea to clear variables as you finish using them, or to restart *Mathematica* after saving important. These actions help you to avoid unintentionally using a variable name that has previously been given a value, and future operations can be performed without risk that previous definitions and assigned constants will interfere with the processing. Clearing variables is done with the **Clear[]** command.

Mathematica is a computer language; it cannot read your mind. *Mathematica* does follow your instructions to the letter. This means that you need to learn to communicate your needs to *Mathematica* in a way that it understands. Here are some basic symbols and other fundamentals.

Assignments are made with either the = or : = symbols. Using = gives the value of the right hand side (RHS) to the name on the left hand side (LHS). Here is an example.

```
x = 2^3
```

8

```
x
```

8

Ending a *Mathematica* command with a **semicolon** (;) performs the command and suppresses the output display. Thus:

```
x = 5;
```

produces no output, but the variable x has still been assigned the value 5.

```
x
```

5

The := symbol is used when you want to define a function which has yet to be evaluated. So defining the function f as

```
Clear[x];
f[x] = x^3
```

x^3

will not use the rule for other variables:

```
f[y]
```

f[y]

Note the difference in the output when the := symbol is used in concert with a variable name x_:

```
f[x_] := x^3;
f[y]
```

y^3

In doing calculations, you will often need to use previous results that you have got. In *Mathematica*, a single **percentage sign** % always stands for your last result. The double percentage sign %% refers to the second previously executed command, the triple percentage sign %%% refers to the third, and so on. The following demonstrates the use of percentage signs.

```
11;

8 + %

19

% * %%

209

(%%% - %) / %%

  198
- ---
   19

(% * 10) + 55

  935
- ---
   19
```

If we change the initial value of 11 to say 6 and execute, *Mathematica* does not automatically adjust other terms even if those statements contain a reference to the changed variable. The only means of re-evaluating terms is by re-executing them and the order in which statements are executed determines the final results.

If input/output names are shown, you can also reference any previous output as %n, where n is the number of the output line. This operation must be done with care, since the output number may easily be changed if you repeat a sequence of operations.

Mathematica is **case sensitive**-- that is, the names x and X are different, pi and Pi are not the same.

Set braces { } enclose unordered sets of objects or ordered lists of objects.

Square brackets [ ] enclose the arguments to *Mathematica* functions.

Parentheses ( ) alter the order of operations in *Mathematica* commands.

Parentheses with asterisks (* *) enclose material (such as comments) in input cells which will not be evaluated.

The question mark ? is used to get help on commands.

The **standard mathematical functions** are denoted by their standard names: +(plus) , - (minus) , * (times) , / (divided by) , ^ (raised to the power) , **`Sin`**, **`Cos`**, **`Tan`**, **`Abs`**, and **`Sqrt`**.

Common constants are predefined in *Mathematica*, their names are: **`Pi`**, **`E`**, **`I`**, and **`Infinity`**. (Yes, they have to be capitalized exactly like this.) These symbols may be entered from the Basic Math Assistant palette as π, e, i, and ∞, and are given in output in those forms.

Unless specified otherwise, all names are assumed to be complex-valued. This is not often a problem, but there are times when this can lead to surprising results.

Let's get the idea of how *Mathematica* works by letting it perform some arithmetic.

In the following lines, use paper and pencil to first predict what you think *Mathematica* will do. Then execute the command and see what actually happens!

```
x = 4 * 6 + 12 / 6 - 1
```

25

```
(-3)^3
```

-27

```
Abs[%]
```

27

```
Pi
```

π

```
v = Sin[Pi / 4]
```

$$\frac{1}{\sqrt{2}}$$

```
w = v^2
```

$$\frac{1}{2}$$

```
v
```

$$\frac{1}{\sqrt{2}}$$

```
Tan[-Pi / 2]
```

ComplexInfinity

```
3 / (5 - Sqrt[x])
```

Power::infy: Infinite expression $\frac{1}{0}$ encountered. ≫

ComplexInfinity

Now enter some commands of your own! Go back to the tangent calculation above and change it to compute the tangent of $\pi/3$.

Now for some slightly more impressive computations. Note what the addition of **//N** does to a command.

```
2^100
```

1 267 650 600 228 229 401 496 703 205 376

```
2^100 // N
```

1.26765×10^{30}

```
N[Pi, 250]
```

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803 ⁞
 82534211706798214808651328230664709384460955058223172535940812848111745028410270193852 ⁞
 05559644622948954930381964428810975665933446128475648233786783165271201909
```

```
400!
```

```
64 034 522 846 623 895 262 347 970 319 503 005 850 702 583 026 002 959 458 684 445 942 802 397 169 186 831 ⁞
 436 278 478 647 463 264 676 294 350 575 035 856 810 848 298 162 883 517 435 228 961 988 646 802 997 937 ⁞
 341 654 150 838 162 426 461 942 352 307 046 244 325 015 114 448 670 890 662 773 914 918 117 331 955 996 ⁞
 440 709 549 671 345 290 477 020 322 434 911 210 797 593 280 795 101 545 372 667 251 627 877 890 009 349 ⁞
 763 765 710 326 350 331 533 965 349 868 386 831 339 352 024 373 788 157 786 791 506 311 858 702 618 270 ⁞
 169 819 740 062 983 025 308 591 298 346 162 272 304 558 339 520 759 611 505 302 236 086 810 433 297 255 ⁞
 194 852 674 432 232 438 669 948 422 404 232 599 805 551 610 635 942 376 961 399 231 917 134 063 858 996 ⁞
 537 970 147 827 206 606 320 217 379 472 010 321 356 624 613 809 077 942 304 597 360 699 567 595 836 096 ⁞
 158 715 129 913 822 286 578 579 549 361 617 654 480 453 222 007 825 818 400 848 436 415 591 229 454 275 ⁞
 384 803 558 374 518 022 675 900 061 399 560 145 595 206 127 211 192 918 105 032 491 008 000 000 000 000 ⁞
 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 ⁞
 000 000 000
```

```
400! // N
```

$$6.403452284662390 \times 10^{868}$$

```
Clear[x];
```

```
f = Sin[2 * x]
```

```
Sin[2 x]
```

```
g = 4 * x^2
```

$$4\,x^2$$

```
h = .25 * Cos[8 * x]
```

```
0.25 Cos[8 x]
```

```
p = f * g
```

$$4\,x^2 \sin[2\,x]$$

```
u1 = f * h
```

```
0.25 Cos[8 x] Sin[2 x]
```

```
u2 = f + h
```

```
0.25 Cos[8 x] + Sin[2 x]
```

```
Clear[x, f, g, h, p, u1, u2]
```

2.3 Functions and Operators

Built into *Mathematica* are the basic functions, relational operators, and logical operators. They are denoted as follows.

Trigonometric: **Sin[]**, **ArcSin[]**, **Cos[x]**, **ArcCos[]**, **Tan[]**, **ArcTan[]**, **Csc[]**, **ArcCsc[]**, **Sec[]**, **ArcSec[]**, **Cot[]**, **ArcCot[]**

Relational: <, <= (or ≤), >, >= (or ≥), == (equal), != (or ≠) (not equal)

Logical: **And** (**&&**), **Or** (**| |**), **Not** (**!**)

Transcendental: **Log[]** (the natural logarithm function), **Log[b,]**(the logarithm base b) , **Exp[]**(the natural exponential function)

Hyperbolic: **Sinh[]**, **ArcSinh[]**, **Cosh[x]**, **ArcCosh[]**, **Tanh[]**, **ArcTanh[]**, **Csch[]**, **ArcCsch[]**, **Sech[]**, **ArcSech[]**, **Coth[]**, **ArcCoth[]**

There are many other functions as well.

Functions can be composed using the **Composition** command. For example:

```
Composition[Cos, Sin][x]
```

```
Cos[Sin[x]]
```

```
Composition[Cos, Sin][Pi]
```

```
1
```

Repeated compositions can be performed as shown. The **Nest** command allows you to compose a function with itself easily.

```
Composition[Cos, Sin, Cos][x]
```

```
Cos[Sin[Cos[x]]]
```

```
Exp[Composition[Tan, Sin][x]]
```

$e^{\text{Tan[Sin[x]]}}$

```
Composition[Cos, Cos][x]
```

```
Cos[Cos[x]]
```

```
Nest[Cos, x, 2]
```

```
Cos[Cos[x]]
```

As mentioned in Section 2, assigning names to expressions is done by using the equal (=) or the "colon equal" (: =). Once a name is assigned to an expression, that name can be used anywhere within the notebook when you want to refer to the expression. Consider the following assignment.

```
f = x^2
```

x^2

We can use the **ReplaceAll** command (abbreviated **/.**) to substitute the value of $x = 4$ into the expression for f. The → symbol is input by using the minus sign then the greater than sign.

```
f /. x → 4
```

```
16
```

We can declare *f* to be a function by using the := command in conjunction with the blank (input as an underscore _) after the argument x. The blank tells *Mathematica* to see f as a pattern which would apply to any argument, not just x.

```
Clear[f];
f[x_] := x^2
```

Now we can easily evaluate as we would any function. Consider for example

```
f[3]
f[10]
f[x]
f[y]
f[x + y]
```

```
9
```

```
100
```

$$x^2$$

$$y^2$$

$$(x + y)^2$$

We can easily consider functions with more variables as shown here:

```
g[x_, y_] := x * Sqrt[y]
h[x_, y_, z_] := Abs[Log[x] * Cos[y] * Exp[z]]
```

We then invoke a function by name and respective input variables

```
g[5, 17]
```

$$5\sqrt{17}$$

```
g[5.0, 17]
```

```
20.6155
```

```
g[5, 17] // N
```

```
20.6155
```

```
N[g[5, 17]]
```

```
20.6155
```

where the data type of the input determines the form of the computed result. For example, the function *g(5, 17)* has two integer inputs, 5 and 17, and because of this *Mathematica* performed exact computations. Evaluation of *g(5.0, 17)* demonstrates that floating-point data types 5.0 and 17.0 tell *Mathematica* to perform approximate calculations while using **N** (either as a function or as a postfix **//N**) always results in approximate output. By default, floating-point results are approximated to 6 digits (not decimal places!) but can be adjusted to any positive integer by the **N** command as shown:

```
N[g[5, 17], 4]
```

```
20.62
```

Remember that it is a good idea to clear the contents of variables and functions when you are done:

```
Clear[x, f, g, h]
```

Operations can also be performed by *Mathematica* on **sets** and **lists**. In *Mathematica*, a set is an unordered collection of elements enclosed by {}. Consider the following sets:

```
A = {a, b, b, c}
```

```
{a, b, b, c}
```

```
B = {d, a}
```

```
{d, a}
```

Notice that repeated elements are retained. Duplicates are not automatically removed from sets and the elements of sets are not re-ordered in output since *Mathematica* assumes that there is an order among the elements of a set. To counteract this assumption we can use set operations. The set of elements that are in A or B can be obtained by the **`Union`** operator, the elements in common with two sets can be derived by the **`Intersection`** operator, and set difference can be employed by the **`Complement`** operator. The following demonstrates the use of these three operators.

```
Union[A, A]
```

```
{a, b, c}
```

```
Union[A, B]
```

```
{a, b, c, d}
```

```
Intersection[A, B]
```

```
{a}
```

```
Complement[A, A]
```

```
{}
```

```
Complement[A, B]
```

```
{b, c}
```

Consider the following list of numbers:

```
A = {3, 13, 6, 6, 11}
```

```
{3, 13, 6, 6, 11}
```

Any element can be taken from the list by using the list name and element position as shown:

```
A[[1]]
A[[2]]
A[[3]]
A[[4]]
A[[5]]
```

```
3

13

6

6

11
```

The mathematical functions that are built into *Mathematica* are mostly set up to be "listable" so that they act separately on each element of a list. This is, however, not true of all functions in *Mathematica*. Unless you set it up specially, a new function that you introduce may treat lists just as single objects. You can use the **Map** command to apply a function like this separately to each element in a list. The **Map** operator can be used to apply a command to every element of a list. The following demonstrates this feature:

```
Sqrt[A]
```

$$\left\{\sqrt{3}, \sqrt{13}, \sqrt{6}, \sqrt{6}, \sqrt{11}\right\}$$

```
Sqrt[A] // N
```

```
{1.73205, 3.60555, 2.44949, 2.44949, 3.31662}
```

```
g[A]
```

```
g[{3, 13, 6, 6, 11}]
```

```
Map[g, A]
```

```
{g[3], g[13], g[6], g[6], g[11]}
```

It will often be useful to take a *Mathematica* expression apart. Suppose that we wish to solve the equation $x^2 = 2$.

```
soln = Solve[x^2 == 2, x]
```

$$\left\{\left\{x \to -\sqrt{2}\right\}, \left\{x \to \sqrt{2}\right\}\right\}$$

We might need to use either of these solutions in future calculations. The first solution is

```
soln[[1]]
```

$$\left\{x \to -\sqrt{2}\right\}$$

so we could plug it into a function using the **/.** command (notice that the form of **soln[[1]]** matches what we need for this operation.)

```
f[x_] := x^3 - 7 * x + 3
```

```
f[x] /. soln[[1]]
```

$$3 + 5\sqrt{2}$$

Notice however that simply plugging **soln[[1]]** into the function *f* does not work.

```
f[soln[[1]]]
```

$$\left\{3 - 7\left(x \to -\sqrt{2}\right) + \left(x \to -\sqrt{2}\right)^3\right\}$$

To extract only the number $-\sqrt{2}$ we take

```
soln[[1, 1, 2]]
```

$$-\sqrt{2}$$

and compute

```
f[soln[[1, 1, 2]]]
```

$$3 + 5\sqrt{2}$$

We could also evalaute *f* at the other solution using either method. Note how the commands change.

```
f[x] /. soln[[2]]
```

$$3 - 5\sqrt{2}$$

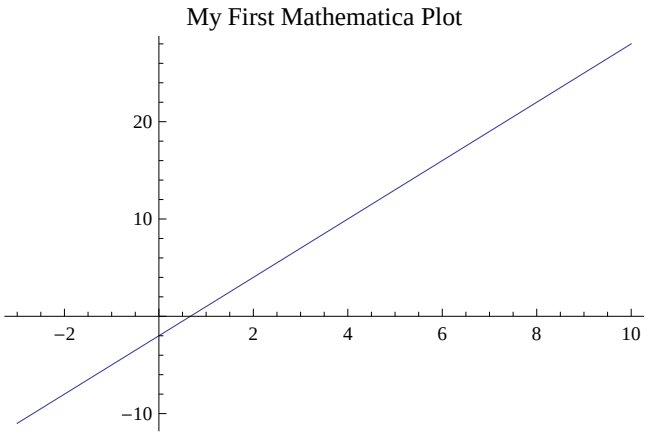
```
f[soln[[2, 1, 2]]]
```

$$3 - 5\sqrt{2}$$

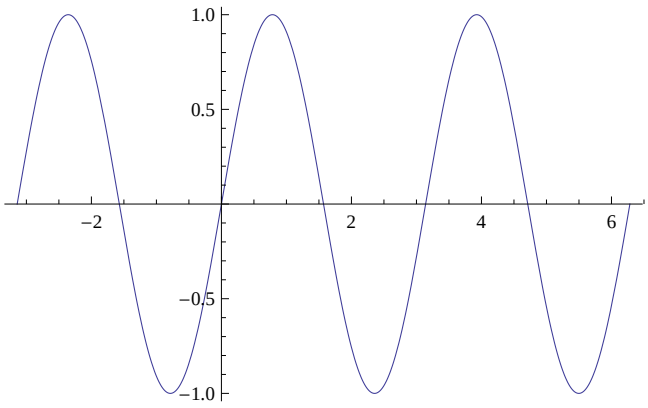
2.4 Plotting with *Mathematica*

Graphs of functions are produced by the **Plot** command. In its simplest form, **Plot** needs to know the function to be plotted and the range of values for the independent variable. Note that **{t,a,b}** is *Mathematica's* way of saying both that the independent variable is *t* and that the plotting interval is [*a*,*b*]. Observe that titles are placed on some of these graphs. Again, try to predict the output before tapping the return key!

```
Plot[3 t - 2, {t, -3, 10}, PlotLabel → "My First Mathematica Plot"]
```




```
f[x_] := Sin[2 x];
(* Here is a sine function! *)
Plot[f[x], {x, -π, 2 π}]
```



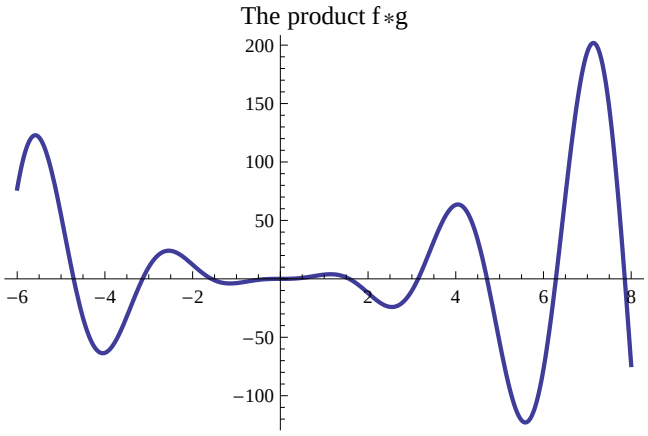
The preceding command has a remark attached. Everything between the (* and *) signs is non-execuable. Notice that ending the **Plot** command with a semicolon suppresses output to the screen.

```
Plot[f[x], {x, -π, 2 π}];
```

```
g[x_] := 4 * x^2;
p[x_] := f[x] * g[x];
```

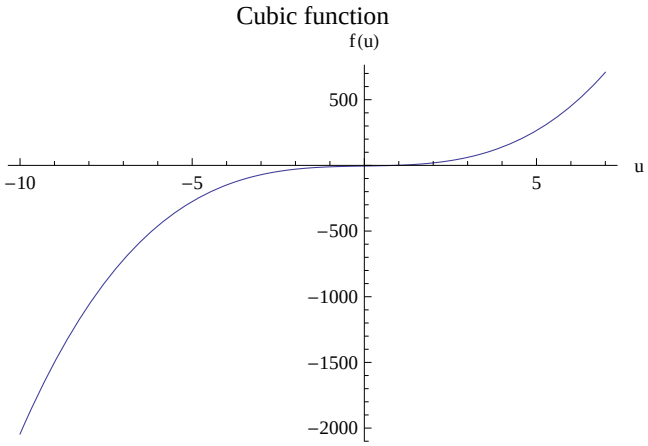
The style of the plot can be changed in various ways, as in the following example.

```
Plot[p[x], {x, -6, 8}, PlotLabel → "The product f*g", PlotStyle → Thick]
```



Consider the following plot.

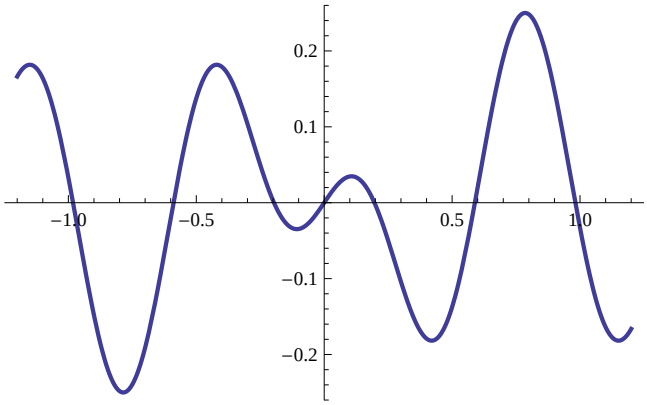
```
Plot[2 u^3 + 4 u - 5, {u, -10, 7}, PlotLabel → "Cubic function", AxesLabel → {"u", "f(u)"}]
```



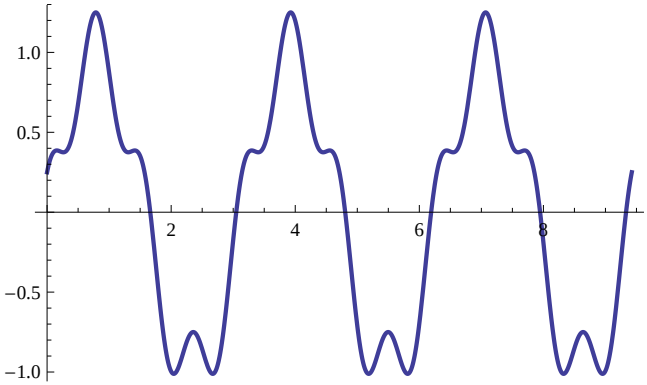
Where does this cubic function cross the *u*-axis? Maybe it would be helpful to cut down the plotting interval from [-10, 7] to [-1, 3], or even narrower, say to [0, 1.5]. Try it. Can you find the *u*-intercept to 2-decimal point accuracy by this process? It is a powerful method, which we call ZOOMING IN.

```
h[x_] := .25 * Cos[8 * x];
u1[x_] := h[x] * f[x];
u2[x_] := f[x] + h[x];
```

```
Plot[u1[x], {x, -1.2, 1.2}, PlotStyle → Thick]
```




```
Plot[u2[x], {x, 0, 3 π}, PlotStyle → Thick]
```

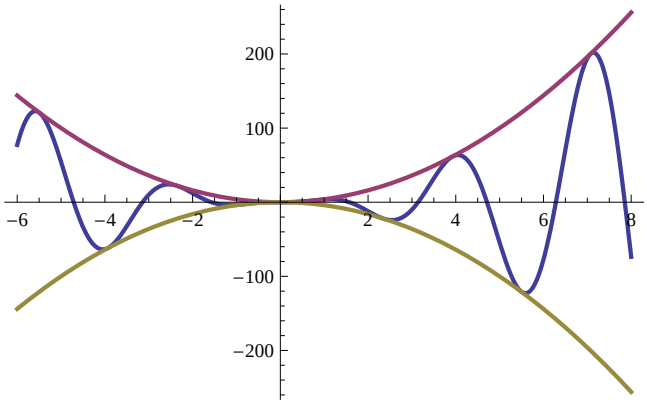


Can you explain why the graphs of *u1* and *u2* look the way they do?

You probably now have a big pile of plots cluttering up your screen. Plots can be reduced in size by selecting the plot, putting the cursor on one of the dots at the corners of the graph, and dragging the cursor on the double arrow that appears. To permanently get rid of plots that you no longer need, click on the cell bracket containing only the plot and depress the delete button. You can delete all of the output in a notebook by using the Delete All Output option under the Cell menu bar.

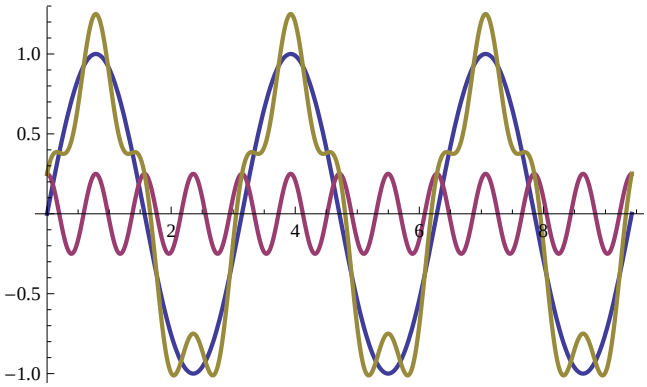
We can also plot many curves simultaneously.

```
Plot[{p[x], g[x], -g[x]}, {x, -6, 8}, PlotStyle → Thick]
```



Can you identify the three different plots? What about the next one? Can you explain the little bumps that appear inside the large dips in the graph of *u2*?

```
Plot[{f[x], h[x], u2[x]}, {x, 0, 3 π}, PlotStyle → Thick]
```

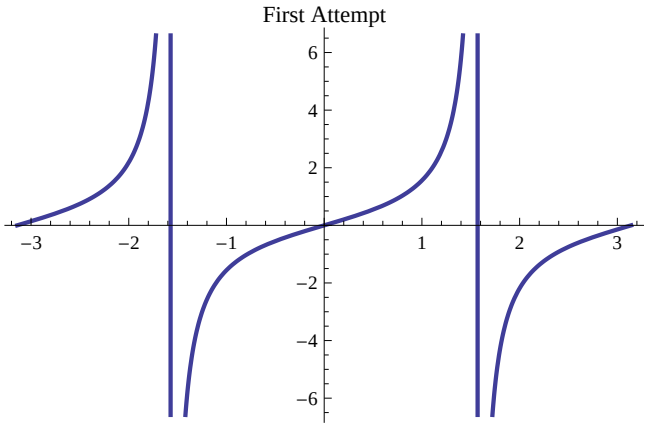


The tangent function is well-known to all of us.

```
f[x_] := Tan[x];
```

Note: This definition completely replaces the previous definition of f . Recall that the graph of $y = \tan(x)$ has vertical asymptotes at all odd multiples of $\pi/2$. Let's see how *Mathematica* handles this.

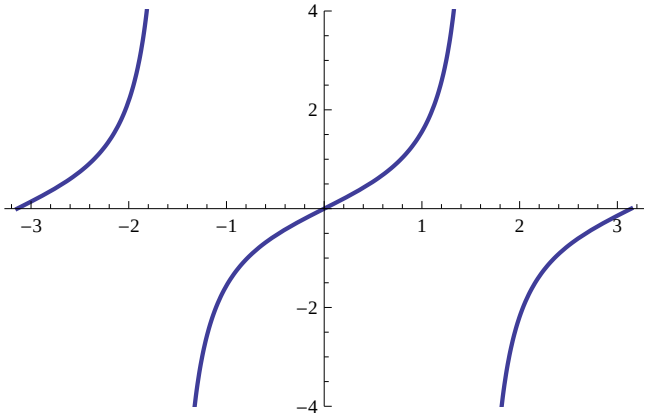
```
Plot[f[x], {x, -π, π}, PlotLabel → "First Attempt", PlotStyle → Thick]
```



Is this what you expected to see? Why does the graph look like this? What can be done to improve the appearance of the plot?

There are two problems. First, we have to tell *Mathematica* that this function is discontinuous. But, there is more. Look at the labels on the vertical axis. The units are larger on the y axis; we want to see the details on a much finer scale. Let's limit the vertical scale to the range from -4 to 4. The optional argument **`PlotRange`** to **`Plot`** explicitly tells *Mathematica* what vertical limits to use on the graph. To handle the discontinuities at $-\pi/2$ and $\pi/2$, we will use another optional argument called **`Exclusions`** to remove those points from the plot.

```
Plot[f[x], {x, -π, π}, Exclusions → {-π / 2, π / 2}, PlotRange → {-4, 4}, PlotStyle → Thick]
```



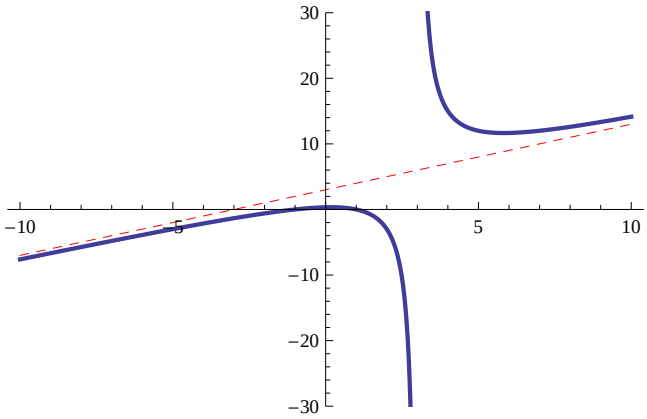
This technique is very useful for any function that has vertical asymptotes.

OK. Now it's your turn. Add appropriate optional arguments to the following command to produce a graph from which you can identify the local minimum. (Don't forget the title.)

```
Plot[(10 - 4 t^3) / (1 - t^2), {t, -3, 5}, PlotStyle → Thick]
```

Now, one last challenge. Use your graph to estimate the (oblique) asymptote for this function. Check your estimate by graphing the function and the asymptote in the same plot. (Here's an example of what we are seeking, and to show how plots look in a notebook. Note the use of the **PlotRange**, **PlotStyle** and **AxisOrigin** options.)

```
g1 = Plot[(1 - t^2) / (3 - t), {t, -10, 10}, Exclusions → {3}, PlotRange → {-30, 30}, PlotStyle → Thick];
g2 = Plot[t + 3, {t, -10, 10}, PlotRange → {-30, 30}, PlotStyle → {Thin, Dashed, Red}];
Show[g1, g2, PlotRange → {{-10, 10}, {-30, 30}}, AxesOrigin → {0, 0}]
```



These examples only scratch the surface of *Mathematica's* abilities when it comes to plotting. *Mathematica* can also produce 3-D plots, plots of parametric curves and surfaces, and animated sequences. We will examine some of these in the next section about *Mathematica* packages. The best general source of information is the on-line help for the **Plot** command and related packages, which can be accessed through the Help menu or by using the command **?Plot**.

2.5 Programming with *Mathematica*

We will create some simple programs in *Mathematica* for use in this manual. The programs will use *repetitive loops* (for loops) and the use of *conditionals* (if - then). There are far more advanced and efficient means to create programs in *Mathematica*, but to introduce them would take us far afield from our task. We call a simple program a *procedure*.

Let's begin with repetitive loops since they will be used often in procedures. The general syntax for a repetitive loop is as follows:

```
For[start, test, incr, body]
```

This command tells *Mathematica* to evaluate the **start** command, the repeatedly evaluate the **body** and **incr** commands until **test** fails. Consider as an example of a repetitive loop which outputs 1, 2 and 3 all raised to the 4th power. The repetitive loop would begin as

```
For[n = 1, n ≤ 3
```

```
Syntax::bktmcp: Expression "For[n = 1, n ≤ 3" has no closing "]".

Syntax::sntxi: Incomplete expression; more input is needed.
```

Since the line is not complete, *Mathematica* responds with a warning. The warning statement(s) disappear when the program is valid and complete. Our repetitive loop and output could be in either of the following forms.

```
For[n = 1, n ≤ 3, n++, Print[n^4]];
1

16

81
```

```
For[n = 1, n ≤ 3, n += 1, Print[n^4]];
1

16

81
```

The second command allows us control over the amount of the increment. Additional lines can be inserted within the body by separting the commands with a semicolon. For example,

```
For[n = 1, n ≤ 3, n += 1,
  Print[n];
  Print[n^4]];
1

1

2

16

3

81
```

Notice that inserting returns can make the procedure easier to read.

The general syntax for a conditional statement is as follows:

```
If[condition, t, f]
```

where the command gives **t** if **condition** evaluates to `True` and **f** if it evalautes to `False`. For more options, the **Which** command is used:

Which[$test_1$, $value_1$, $test_2$, $value_2$, …]

where the command evaluates each of the $test_i$ in turn, returning the value of $value_i$ corresponding to the first of the $test_i$ which yields `True`.

As an example of the use of a conditional statement, let's pick 1000 random numbers between 0 and 2. We will say the number is good if it is less than one. We will multiply all those between 0 and 1 together and keep a record of how many between 0 and 1 are selected. We will use the random number generator of *Mathematica* to select the random number. Let's see some of the random numbers generated by this procedure.

```
For[n = 1, n ≤ 5, n++, Print[RandomReal[{0, 2}]]];
```

```
1.97746

0.307963

0.254608

0.332647

1.31522
```

```
good = 0;
bad = 0;
prod = 1;
For[n = 1, n ≤ 100, n++, x = RandomReal[{0, 2}];
  If[x < 1, good = good + 1; prod = prod x, bad = bad + 1];];
Print["Less than one= ", good];
Print["Product= ", prod];
```

```
Less than one= 49

Product= 9.43247 × 10^-23
```

Note that the product is very small as it should be. You can try 1000 numbers and see what the product becomes and see if 1/2 of the numbers chosen are between 0 and 1.

```
Clear[x, good, bad, prod];
```

Finally let's examine more compliacted procedures. We can combine several activities together and call it a procedure. Then whenever we wish to use it we just have to call for it. The general syntax of a *Mathematica* procedure is as follows:

```
Procedure_Name[input_variables] := {
   statement;
     statement;
     ....
    statement};
```

A function is an example of a procedure. We can program the function $f(x) = x^2$ as a procedure as follows:

```
f[x_] := x^2;
```

The procedure can now be invoked by name, in this case *f*, as demonstrated below.

```
f[2]
f[4]
f[a]
f[b]
f[a + b]
```

```
4
```

```
16
```

$$a^2$$

$$b^2$$

$$(a + b)^2$$

In constructing more complicated procedures, it is often useful to declare the scope of the variables used. A local variable is active only within the procedure where it is declared while a variable declared as global retains its value for the entire notebook. The **Module** structure is used to declare local variables. Consider this distinction within the following procedure which computes the sum of square integers from 1 to 3.

```
Clear[n];
SumOfSquares[] :=
  Module[{n}, totalsum = 0;
   For[n = 1, n ≤ 3, n++,
    totalsum = totalsum + n^2];
   Return[totalsum]
  ];
```

We invoke this procedure by name,

```
SumOfSquares[]
```

```
14
```

The incremental variable **n** was declared local by the programmer. In comparison, the variable **totalsum** was by default declared global. Because of this distinction, **totalsum** retains its value outside of this procedure, while variable **n** does not.

```
totalsum
```

```
14
```

```
n
```

```
n
```

3. Using *Mathematica* in Calculus and Differential Equations

In this second introductory section we will give demonstrations of how *Mathematica* can be used in calculus and differential equations. Later, as you work through some of the lab sections, it may be helpful to return to this section to see how some of the code in *Mathematica* is actually used. We will begin with some calculus operations that include algebraic manipulations.

3.1 Calculus Examples

In this section we will see how to differentiate, integrate, take limits, and manipulate algebraic expressions with *Mathematica*.

Consider the following sequence of operations relating to the function $x(t) = e^{-\alpha t}\cos(\omega t + \theta)$. What information is obtained in each step?

```
x = Exp[-α * t] * Cos[ω * t + θ];
```

```
dx = D[x, t]
```

$$-e^{-t\,\alpha}\,\alpha\,\text{Cos}[\theta + t\,\omega] - e^{-t\,\alpha}\,\omega\,\text{Sin}[\theta + t\,\omega]$$

```
dx2 = D[x, {t, 2}]
```

$$e^{-t\,\alpha}\,\alpha^2\,\text{Cos}[\theta + t\,\omega] - e^{-t\,\alpha}\,\omega^2\,\text{Cos}[\theta + t\,\omega] + 2\,e^{-t\,\alpha}\,\alpha\,\omega\,\text{Sin}[\theta + t\,\omega]$$

```
Factor[dx]
```

$$-e^{-t\,\alpha}\,(\alpha\,\text{Cos}[\theta + t\,\omega] + \omega\,\text{Sin}[\theta + t\,\omega])$$

```
critpt = FullSimplify[Solve[dx == 0, t]]
```

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

$$\left\{\left\{t \to -\frac{\text{ArcCos}\left[-\frac{\sqrt{(\omega\,\text{Cos}[\theta]-\alpha\,\text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]}{\omega}\right\}, \left\{t \to \frac{\text{ArcCos}\left[-\frac{\sqrt{(\omega\,\text{Cos}[\theta]-\alpha\,\text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]}{\omega}\right\},\right.$$
$$\left.\left\{t \to -\frac{\text{ArcCos}\left[\frac{\sqrt{(\omega\,\text{Cos}[\theta]-\alpha\,\text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]}{\omega}\right\}, \left\{t \to \frac{\text{ArcCos}\left[\frac{\sqrt{(\omega\,\text{Cos}[\theta]-\alpha\,\text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]}{\omega}\right\}\right\}$$

```
FullSimplify[dx2 /. critpt[[2]]]
```

$$e^{-\frac{\alpha\,\text{ArcCos}\left[-\frac{\sqrt{(\omega\,\text{Cos}[\theta]-\alpha\,\text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]}{\omega}}\left((\alpha-\omega)\,(\alpha+\omega)\,\text{Cos}\left[\theta + \text{ArcCos}\left[-\frac{\sqrt{(\omega\,\text{Cos}[\theta]-\alpha\,\text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]\right] + \right.$$
$$\left. 2\,\alpha\,\omega\,\text{Sin}\left[\theta + \text{ArcCos}\left[-\frac{\sqrt{(\omega\,\text{Cos}[\theta]-\alpha\,\text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]\right]\right)$$

Since the second derivative at the critical point will be negative (for real values of ω, α, and θ), this critical point must be a (local) maximum. The value of the function at this maximum is

```
Simplify[x /. critpt[[2]]]
```

$$e^{-\frac{\alpha \text{ArcCos}\left[-\frac{\sqrt{(\omega \text{Cos}[\theta]-\alpha \text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]}{\omega}} \text{Cos}\left[\theta+\text{ArcCos}\left[-\frac{\sqrt{(\omega \text{Cos}[\theta]-\alpha \text{Sin}[\theta])^2}}{\sqrt{\alpha^2+\omega^2}}\right]\right]$$

Indefinite and definite integration is obtained in a natural way.

```
Integrate[x, t]
```

$$\frac{e^{-t\alpha}\,(-\alpha \text{Cos}[\theta+t\,\omega]+\omega \text{Sin}[\theta+t\,\omega])}{\alpha^2+\omega^2}$$

Note that the constant of integration is not included in these results.

Definite integrals are very similar. Here's an example taken from the CRC Tables of Integrals.

```
Integrate[u / Sin[u], {u, 0, π / 2}]
```

```
2 Catalan
```

```
% // N
```

```
1.83193
```

For further illustrations, consider the following two improper integrals.

```
Integrate[Exp[-u^2], {u, 0, ∞}]
```

$$\frac{\sqrt{\pi}}{2}$$

```
Integrate[Exp[-a * u^2], {u, 0, ∞}]
```

$$\text{If}\left[\text{Re}[a]>0,\ \frac{\sqrt{\pi}}{2\sqrt{a}},\ \text{Integrate}\left[e^{-a\,u^2},\ \{u,\ 0,\ \infty\},\ \text{Assumptions}\to \text{Re}[a]\le 0\right]\right]$$

Note that *Mathematica* is unable to evaluate this limit until something is known about the parameter *a*.

```
Assuming[a > 0, Integrate[Exp[-a * u^2], {u, 0, ∞}]]
```

$$\frac{\sqrt{\pi}}{2\sqrt{a}}$$

Here is a simple limit example. Notice that the limit is being taken from the right.

```
Limit[Tan[θ], θ → π / 2]
```

```
-∞
```

Limits from the left are also available.

```
Limit[Tan[θ], θ → π / 2, Direction → 1]
```

$$\infty$$

■ An Application: Integration of a Rational Function

Consider the proper rational function:

```
Clear[x]
```

```
f = (x^2 - 26 * x - 47) / (x^5 + 5 * x^3 + 3 * x^4 + 11 * x^2 - 20)
```

$$\frac{-47 - 26\,x + x^2}{-20 + 11\,x^2 + 5\,x^3 + 3\,x^4 + x^5}$$

The antiderivative of *f* can be obtained by a variety of different means. Let's examine a few and compare the information obtained in each case.
The simplest, but least instructive, approach is to simply let *Mathematica* do all the work. Note that the constant of integration is not included in indefinite integrals.

```
Integrate[f, x]
```

$$\frac{1}{270}\left(\frac{90}{2+x} + 182\sqrt{5}\,\text{ArcTan}\left[\frac{x}{\sqrt{5}}\right] - 360\,\text{Log}[-1+x] + 230\,\text{Log}[2+x] + 65\,\text{Log}\left[5+x^2\right]\right)$$

To emphasize integration techniques while still avoiding algebraic complications, we can utilize *Mathematica* to determine the partial fraction expansion of the integrand.

```
fpf = Apart[f]
```

$$-\frac{4}{3\,(-1+x)} - \frac{1}{3\,(2+x)^2} + \frac{23}{27\,(2+x)} + \frac{13\,(7+x)}{27\left(5+x^2\right)}$$

From this partial fraction decomposition it is now a relatively simple matter to compute the integral BY HAND. The result can be compared with the previous answer. But let *Mathematica* do it.

```
Integrate[fpf, x]
```

$$\frac{1}{270}\left(\frac{90}{2+x} + 182\sqrt{5}\,\text{ArcTan}\left[\frac{x}{\sqrt{5}}\right] - 360\,\text{Log}[-1+x] + 230\,\text{Log}[2+x] + 65\,\text{Log}\left[5+x^2\right]\right)$$

The third and final approach to this problem will be the most detailed. The purpose here is to emphasize the process of finding a partial fraction expansion without fear of unmanageable algebra. The key here is to enter the correct form for the partial fraction decomposition. This form depends upon the irreducible factors in the denominator of *f*.

```
Factor[Denominator[f]]
```

$$(-1+x)\,(2+x)^2\left(5+x^2\right)$$

The appropriate general form for the decomposition will be

```
form = a / (x - 1) + b / (x + 2) + c / (x + 2)^2 + (d + e * x) / (x^2 + 5)
```

$$\frac{a}{-1+x} + \frac{c}{(2+x)^2} + \frac{b}{2+x} + \frac{d+e\,x}{5+x^2}$$

The second step is to determine the (linear) equations that define the constants *a*, *b*, *c*, *d*, and *e*.

```
eqn = f * Denominator[f] == Collect[Factor[form * Denominator[f]], x]
```

$$-47 - 26\,x + x^2 == 20\,a - 10\,b - 5\,c - 4\,d + (20\,a + 5\,b + 5\,c - 4\,e)\,x + (9\,a + 3\,b - c + 3\,d)\,x^2 + (4\,a + b + c + d + 3\,e)\,x^3 + (a + b + e)\,x^4$$

This equation will be satisfied (for all real *x*, except 1 and -2) for any solution to the following system of five equations with five unknowns.

```
eq1 = a + b + e == 0;
eq2 = 4 * a + b + c + d + 3 * e == 0;
eq3 = 9 * a + 3 * b - c + 3 * d == 1;
eq4 = 20 * a + 5 * b + 5 * c - 4 * e == -26;
eq5 = 20 * a - 10 * b - 5 * c - 4 * d == -47;
```

The solution to these equations is

```
coeffs = Solve[{eq1, eq2, eq3, eq4, eq5}, {a, b, c, d, e}]
```

$$\left\{\left\{a \to -\frac{4}{3},\ b \to \frac{23}{27},\ c \to -\frac{1}{3},\ d \to \frac{91}{27},\ e \to \frac{13}{27}\right\}\right\}$$

Substituting this result, which is a set, into the original form for the partial fraction decomposition yields

```
form /. coeffs
```

$$\left\{-\frac{4}{3\,(-1+x)} - \frac{1}{3\,(2+x)^2} + \frac{23}{27\,(2+x)} + \frac{\frac{91}{27} + \frac{13\,x}{27}}{5 + x^2}\right\}$$

Let's confirm the answer.

```
Simplify[%]
```

$$\left\{\frac{-47 - 26\,x + x^2}{(-1+x)\,(2+x)^2\left(5+x^2\right)}\right\}$$

The integral is now much simpler to evaluate by hand, but here it is one last time according to *Mathematica*.

```
Integrate[%%, x]
```

$$\left\{\frac{1}{270}\left(\frac{90}{2+x} + 182\sqrt{5}\,\text{ArcTan}\left[\frac{x}{\sqrt{5}}\right] - 360\,\text{Log}[-1+x] + 230\,\text{Log}[2+x] + 65\,\text{Log}\left[5+x^2\right]\right)\right\}$$

Note that this same outline can be used for ANY partial fraction problem. The key steps are i) knowing the correct form for the decomposition and ii) understanding how to identify the conditions that the coefficients must satisfy. These are (relatively) high level concepts; the user is relieved of the rather complicated mechanical manipulations.

3.2 Differential Equation Demonstrations

In this section we will demonstrate the use of *Mathematica* in working with differential equations. The following topics are covered in the following subsections.

- Direction Fields and Phase Plane Analysis (Sections 1.3 and 5.4 of the Nagle/Saff/Snider text)
- Symbolic Solutions to Ordinary Differential Equations and Initial-Value Problems (Chapter 4 of the Nagle/Saff/-Snider text)
- Expressions and Functions (Chapter 4 of the Nagle/Saff/Snider text)
- Systems of Ordinary Differential Equations (Sections 5.2, 5.4 and 5.5 of the Nagle/Saff/Snider text)
- Numeric Solutions (Sections 3.6 and 5.3 of the Nagle/Saff/Snider text)
- Series Solutions (Chapter 8 of the Nagle/Saff/Snider text)
- Laplace Transforms (Chapter 7 of the Nagle/Saff/Snider text)

■ Direction Fields and Phase Plane Analysis (Sections 1.3 and 5.4 of the Nagle/Saff/Snider text)

Many introductory courses begin by trying to develop the student's understanding of what a differential equation is, what it means for a function to solve an ODE, and how to perform some analysis directly from the differential equation. Graphical methods are commonly employed in these discussions. The *Mathematica* **`Plot`** command and the **`VectorPlot`** command from the VectorFieldPlots package provide a graphical interface.

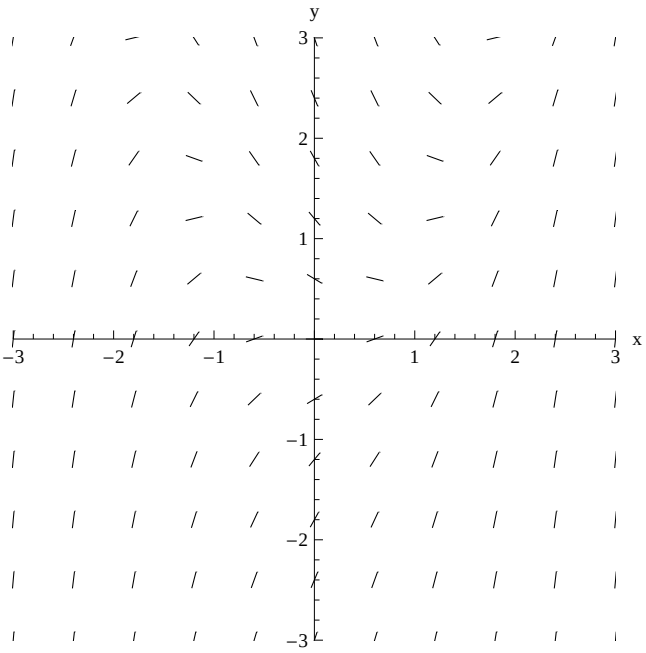
To begin, consider the (linear) differential equation

$$\frac{dy}{dx} = x^2 - y$$

Note that this is the first example used in Section 1.3 of Nagle, Saff, and Snider. The next few commands reproduce a few of the figures displayed on pp.16-17.

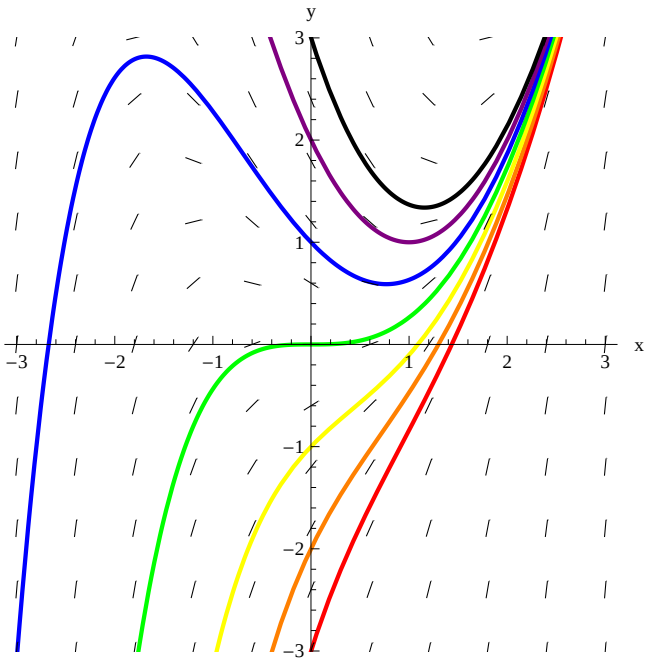
We use **`VectorPlot`** to display a direction field for a differential equation as follows. A range for the independent and dependent variables must be included. The options are used to make the direction field look like those in the text; you can use the Help menu to investigate what they do.

```
dir = VectorPlot[{1, x^2 - y}, {x, -3, 3}, {y, -3, 3}, VectorPoints → 11, Axes → True,
  Frame → False, VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.02, 1, None},
  PlotRange → {{-3, 3}, {-3, 3}}, Axes → True, AxesLabel → {"x", "y"}, AspectRatio → 1]
```



Solution curves through specified points are produced by plotting solutions to the appropriate initial value problems.

```
soln1 = DSolve[{y'[x] == x^2 - y[x], y[0] == -3}, y[x], x];
g1 = Plot[y[x] /. soln1, {x, -3, 3}, PlotRange → {-3, 3}, PlotStyle → {Red, Thick}];
soln2 = DSolve[{y'[x] == x^2 - y[x], y[0] == -2}, y[x], x];
g2 = Plot[y[x] /. soln2, {x, -3, 3}, PlotRange → {-3, 3}, PlotStyle → {Orange, Thick}];
soln3 = DSolve[{y'[x] == x^2 - y[x], y[0] == -1}, y[x], x];
g3 = Plot[y[x] /. soln3, {x, -3, 3}, PlotRange → {-3, 3}, PlotStyle → {Yellow, Thick}];
soln4 = DSolve[{y'[x] == x^2 - y[x], y[0] == 0}, y[x], x];
g4 = Plot[y[x] /. soln4, {x, -3, 3}, PlotRange → {-3, 3}, PlotStyle → {Green, Thick}];
soln5 = DSolve[{y'[x] == x^2 - y[x], y[0] == 1}, y[x], x];
g5 = Plot[y[x] /. soln5, {x, -3, 3}, PlotRange → {-3, 3}, PlotStyle → {Blue, Thick}];
soln6 = DSolve[{y'[x] == x^2 - y[x], y[0] == 2}, y[x], x];
g6 = Plot[y[x] /. soln6, {x, -3, 3}, PlotRange → {-3, 3}, PlotStyle → {Purple, Thick}];
soln7 = DSolve[{y'[x] == x^2 - y[x], y[0] == 3}, y[x], x];
g7 = Plot[y[x] /. soln7, {x, -3, 3}, PlotRange → {-3, 3}, PlotStyle → {Black, Thick}];
Show[g1, g2, g3, g4, g5, g6, g7, dir, AspectRatio → 1, AxesLabel → {"x", "y"}]
```

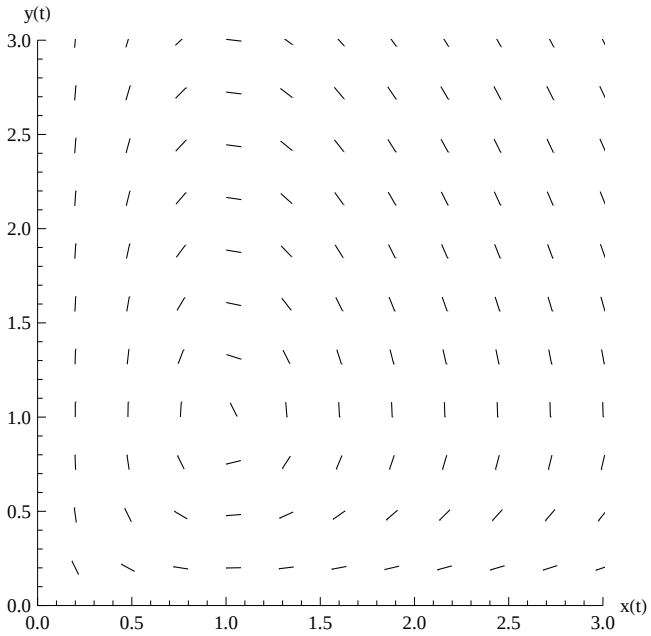


Only slight alterations are needed for higher-order equations, or for systems. To illustrate, consider the predator-prey system

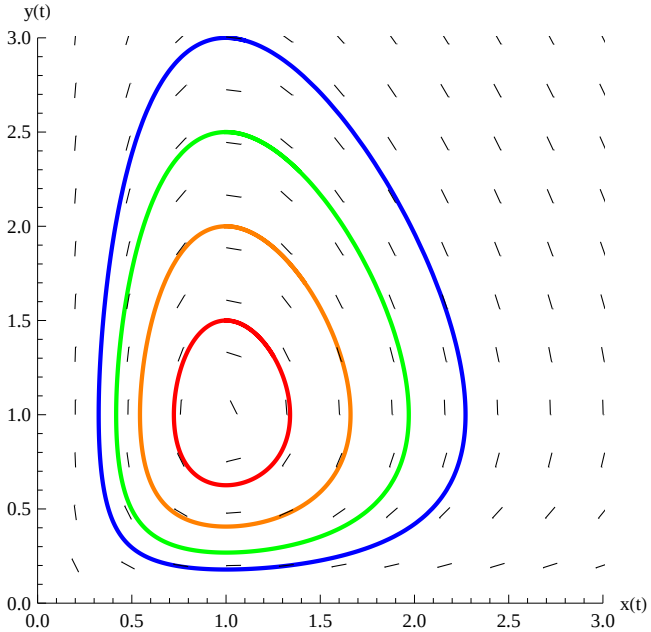
$$\frac{dx}{dt} = x(t)\,y(t) - x(t), \quad \frac{dy}{dt} = 2\,y(t) - 2\,x(t)\,y(t)$$

where *x* denotes the size of the predator population and *y* is the prey.

```
dir = VectorPlot[{x y - x, 2 y - 2 x y}, {x, 0.2, 3}, {y, 0.2, 3}, PlotRange → {{0, 3}, {0, 3}},
  VectorPoints → 11, Axes → True, Frame → False, VectorStyle → {Black, Arrowheads[0]},
  VectorScale → {.02, 1, None}, PlotRange → {{-3, 3}, {-3, 3}}, Axes → True,
  AxesLabel → {"x(t)", "y(t)"}, AxesOrigin → {0, 0}, AspectRatio → 1]
```

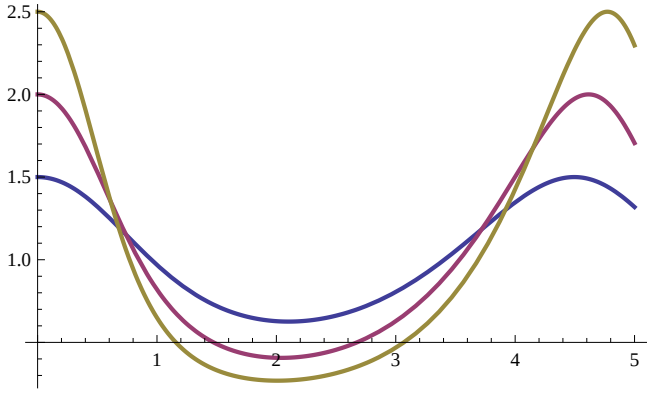


```
soln1 = NDSolve[{x'[t] == x[t] y[t] - x[t],
    y'[t] == 2 y[t] - 2 x[t] y[t], x[0] == 1, y[0] == 3/2}, {x[t], y[t]}, {t, 0, 5}];
g1 = ParametricPlot[{x[t], y[t]} /. soln1, {t, 0, 5},
   PlotStyle → {Red, Thick}, PlotRange → {{0, 3}, {0, 3}}];
soln2 = NDSolve[{x'[t] == x[t] y[t] - x[t], y'[t] == 2 y[t] - 2 x[t] y[t], x[0] == 1, y[0] == 2},
   {x[t], y[t]}, {t, 0, 5}];
g2 = ParametricPlot[{x[t], y[t]} /. soln2, {t, 0, 5},
   PlotStyle → {Orange, Thick}, PlotRange → {{0, 3}, {0, 3}}];
soln3 = NDSolve[{x'[t] == x[t] y[t] - x[t], y'[t] == 2 y[t] - 2 x[t] y[t], x[0] == 1, y[0] == 5/2},
   {x[t], y[t]}, {t, 0, 5}];
g3 = ParametricPlot[{x[t], y[t]} /. soln3, {t, 0, 5},
   PlotStyle → {Green, Thick}, PlotRange → {{0, 3}, {0, 3}}];
soln4 = NDSolve[{x'[t] == x[t] y[t] - x[t], y'[t] == 2 y[t] - 2 x[t] y[t], x[0] == 1, y[0] == 3},
   {x[t], y[t]}, {t, 0, 5}];
g4 = ParametricPlot[{x[t], y[t]} /. soln4, {t, 0, 5},
   PlotStyle → {Blue, Thick}, PlotRange → {{0, 3}, {0, 3}}];
Show[g1, g2, g3, g4, dir, AspectRatio → 1, AxesLabel → {"x(t)", "y(t)"}]
```



Plots of individual solutions can be obtained by using the **`Plot`** command:

```
Plot[{y[t] /. soln1, y[t] /. soln2, y[t] /. soln3}, {t, 0, 5}, PlotStyle → Thick]
```



This is only the beginning of what can be done with these packages. Please consult the Help menu for additional information for a complete description of the available options and examples.

- **Symbolic Solutions to Ordinary Differential Equations and Initial-Value Problems (Chapter 4 of the Nagle/Saff/Snider text)**

While *Mathematica* is quite happy to accept the inputs to its commands in almost any form, it is practically, esthetically, and pedagogically preferable to use descriptive names for the individual parts of a problem. To illustrate, let's find the general solution to

$$x'' + 4x' + 4x = 2te^{-2t}$$

The differential equation can be specified as

```
eqn = D[x[t], {t, 2}] + 4 * D[x[t], t] + 4 * x[t] == 2 * t * Exp[-2 * t]
```

$$4x[t] + 4x'[t] + x''[t] == 2e^{-2t}t$$

and the general solution is found using **DSolve**.

```
gsoln = DSolve[eqn, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{1}{3}e^{-2t}t^3 + e^{-2t}C[1] + e^{-2t}tC[2]\right\}\right\}$$

Note that *Mathematica* introduces the constants `C[1]` and `C[2]` as arbitrary constants. The solution to an initial value problem can be obtained by including initial conditions in a set containing the differential equation. For example, the initial conditions $x(1) = 0$, $x'(1) = 1$ can be implemented as

```
initcond = {x[1] == 0, x'[1] == 1}
```

$$\{x[1] == 0, x'[1] == 1\}$$

Then, the initial value problem is

```
ivp = {eqn, initcond}
```

$$\left\{4x[t] + 4x'[t] + x''[t] == 2e^{-2t}t, \{x[1] == 0, x'[1] == 1\}\right\}$$

and its solution is found using **DSolve**.

```
DSolve[ivp, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{1}{3} e^{-2t} (-1+t) \left(-2+3e^2+t+t^2\right)\right\}\right\}$$

- **Expressions and Functions (Chapter 4 of the Nagle/Saff/Snider text)**

Notice how the output from **DSolve** is a *Mathematica* assignment. While there are specific cases where this is useful, many circumstances call for a solution in another form. The key is to understand the structure of the output from **DSolve**. Let's illustrate with the same example as above, with initial conditions specified, so that the constants refer to the values of the solution and its first derivative at $t = 1$.

```
initcond = {x[1] == c1, x'[1] == c2};
ivp = {eqn, initcond};
gsoln = DSolve[ivp, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{1}{3} e^{-2t} \left(2-3\,c1\,e^2-3\,c2\,e^2-3t+6\,c1\,e^2\,t+3\,c2\,e^2\,t+t^3\right)\right\}\right\}$$

If we want to use the solution as a function instead of as an assignment, we can take that part of the output of **DSolve**.

```
gsoln[[1, 1, 2]]
```

$$\frac{1}{3} e^{-2t} \left(2-3\,c1\,e^2-3\,c2\,e^2-3t+6\,c1\,e^2\,t+3\,c2\,e^2\,t+t^3\right)$$

We can also create a function from that part of the solution and use that function to find the solution passing through (0,1) with unit slope as follows.

```
psoln[t_, c1_, c2_] = DSolve[ivp, x[t], t][[1, 1, 2]]
```

$$\frac{1}{3} e^{-2t} \left(2-3\,c1\,e^2-3\,c2\,e^2-3t+6\,c1\,e^2\,t+3\,c2\,e^2\,t+t^3\right)$$

```
psoln[t, 0, 1]
```

$$\frac{1}{3} e^{-2t} \left(2-3e^2-3t+3e^2\,t+t^3\right)$$

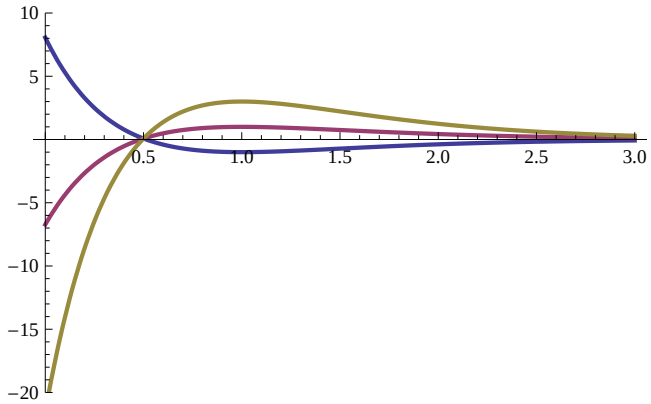
The same calculation can be done using the output of **gsoln** directly:

```
gsoln /. {c1 → 0, c2 → 1}
```

$$\left\{\left\{x[t] \to \frac{1}{3} e^{-2t} \left(2-3e^2-3t+3e^2\,t+t^3\right)\right\}\right\}$$

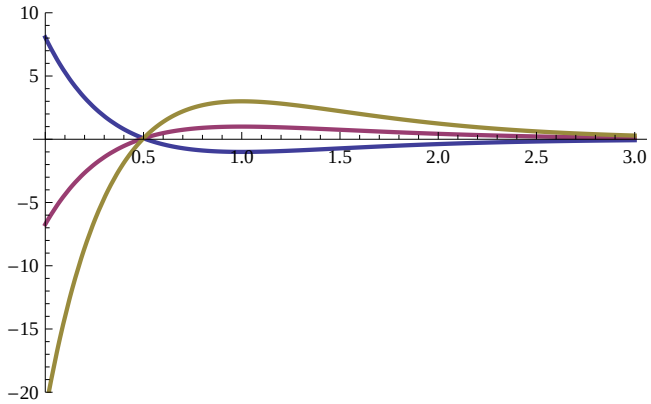
To conclude this discussion, let's plot the solutions that have critical points (i.e., c2=0) at (-1,1), (1,1), and (3,1). We first use the function **psoln** we derived above.

```
Plot[{psoln[t, -1, 0], psoln[t, 1, 0], psoln[t, 3, 0]},
 {t, 0, 3}, PlotRange → {-20, 10}, PlotStyle → Thick]
```



The same result is possible using assignment, but the command is much more involved.

```
Plot[{x[t] /. gsoln /. {c1 → -1, c2 → 0}, x[t] /. gsoln /. {c1 → 1, c2 → 0},
  x[t] /. gsoln /. {c1 → 3, c2 → 0}}, {t, 0, 3}, PlotRange → {-20, 10}, PlotStyle → Thick]
```



Additional information about DSolve is available from the Help menu or by entering **?DSolve**.

■ Systems of Ordinary Differential Equations (Sections 5.2, 5.4 and 5.5 of the Nagle/Saff/Snider text)

Systems of ODEs are analyzed in a completely parallel manner. The system of equations is specified as a list (with or without initial conditions); the dependent variables are also specified as a list. To illustrate, consider the (linearized) model of a pendulum: This second-order equation is
equivalent to the system of first-order equations

```
sys = {D[θ[t], t] == v[t], D[v[t], t] == -3 * θ[t]}
```

```
{θ′[t] == v[t], v′[t] == -3 θ[t]}
```

in terms of the dependent variables

```
fns = {θ[t], v[t]};
```

The general solution is found to be

```
soln = DSolve[sys, fns, t]
```

$$\left\{\left\{\theta[t] \to C[1]\,\text{Cos}\left[\sqrt{3}\,t\right] + \frac{C[2]\,\text{Sin}\left[\sqrt{3}\,t\right]}{\sqrt{3}},\ v[t] \to C[2]\,\text{Cos}\left[\sqrt{3}\,t\right] - \sqrt{3}\,C[1]\,\text{Sin}\left[\sqrt{3}\,t\right]\right\}\right\}$$

Note that **DSolve** returns a list of transformation rules, and the rules are in the order that the functions appear in the list.

Most systems of ODEs do not have explicit solutions. The real power of *Mathematica* for systems of ODEs is seen in its handling of numerical solutions. This procedure will be demonstrated in the next section.

■ Numeric Solutions (Sections 3.6 and 5.3 of the Nagle/Saff/Snider text)

Numeric solutions to initial value problems can be used in a variety of ways in an introductory course, including direction fields, qualitative behavior of solutions, existence theory, and regularity theory. The **NDSolve** command can be used for each of these topics. For ordinary differential equations, **NDSolve** by default uses an LSODA approach, switching between a non-stiff Adams method and a stiff Gear backward differentiation formula method. Notice that there are no built-in *Mathematica* procedures for Euler's method, improved Euler's method and other typical numerical methods discussed in an introductory course. These methods are easily implemented in *Mathematica*, as will be demonstrated in a later section.

To begin our discussion, consider the initial-value problem

```
eqn = D[x[t], t] == x[t]^2 + t;
initcond = x[0] == 0;
ivp = {eqn, initcond};
```

While this problem does have an explicit solution, it cannot be found using methods typically found in an introductory course. But, it is reasonable to ask if a solution does exist for all $t > 0$. Numerical solutions can be used, with caution, to discuss questions of this type.

```
numsoln = NDSolve[ivp, x[t], {t, 0, 1}]
```

```
{{x[t] → InterpolatingFunction[{{0., 1.}}, <>][t]}}
```

The output from **NDSolve** is quite different from that produced by **DSolve**. The output is an InterpolatingFunction object that can be used to compute an approximate solution at a given (numeric) value of t between 0 and 1. For example:

```
numsoln /. t → 0
```

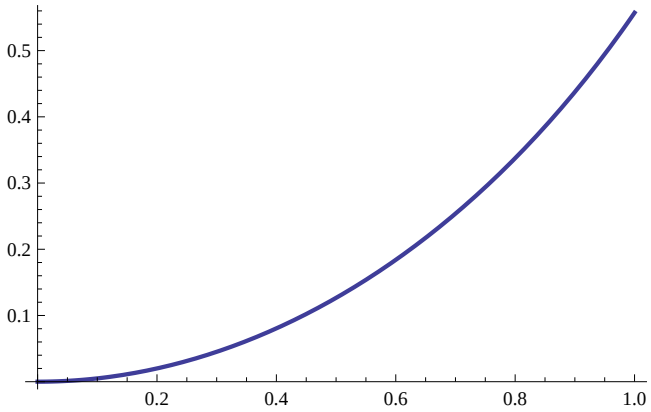
$$\left\{\left\{x[0] \to 2.64956 \times 10^{-25}\right\}\right\}$$

```
numsoln /. t → 1 / 10
```

$$\left\{\left\{x\left[\frac{1}{10}\right] \to 0.00500055\right\}\right\}$$

The plot of a numerical solution can be created using the **Plot** command. For example:

```
Plot[x[t] /. numsoln, {t, 0, 1}, PlotStyle → Thick]
```



The **NDSolve** command works equally well with higher-order equations and with systems. The basic operations for a higher-order problem will be illustrated using the damped nonlinear pendulum. Multiple graphs may be created. (Note how the solution is selected and the derivative of the solution is created using **Evaluate**.)

```
eqn = D[θ[t], {t, 2}] + .2 * D[θ[t], t] + Sin[θ[t]] == 0;
initcond = {θ[0] == π / 4, θ'[0] == 0};
ivp = {eqn, initcond};
```

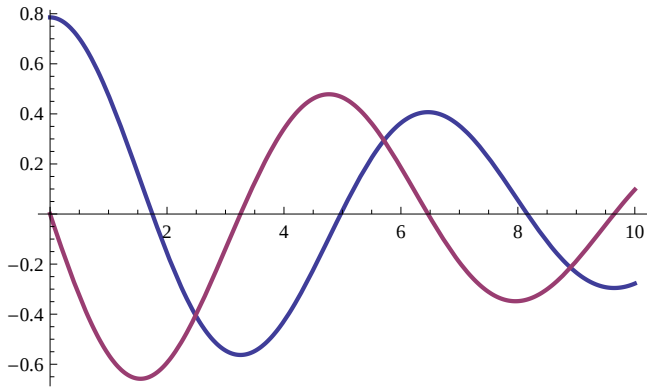
```
numsoln = NDSolve[ivp, θ[t], {t, 0, 10}]
```

```
{{θ[t] → InterpolatingFunction[{{0., 10.}}, <>][t]}}
```

```
dnumsoln = Evaluate[D[numsoln[[1, 1, 2]], t]]
```

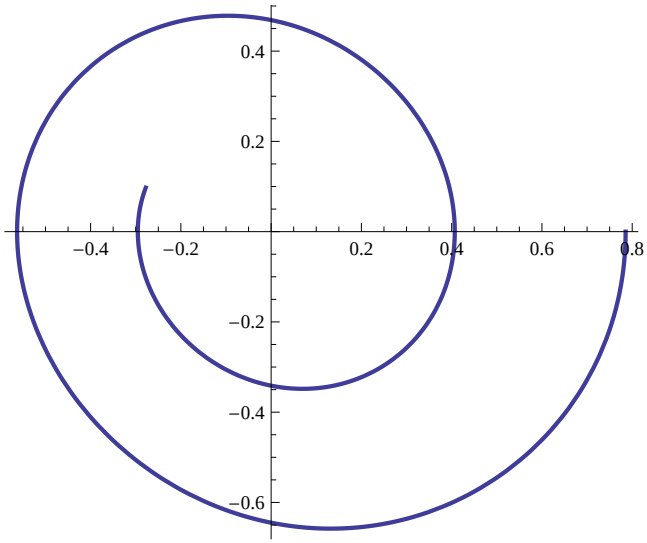
```
InterpolatingFunction[{{0., 10.}}, <>][t]
```

```
Plot[{numsoln[[1, 1, 2]], dnumsoln}, {t, 0, 10}, PlotStyle → Thick]
```



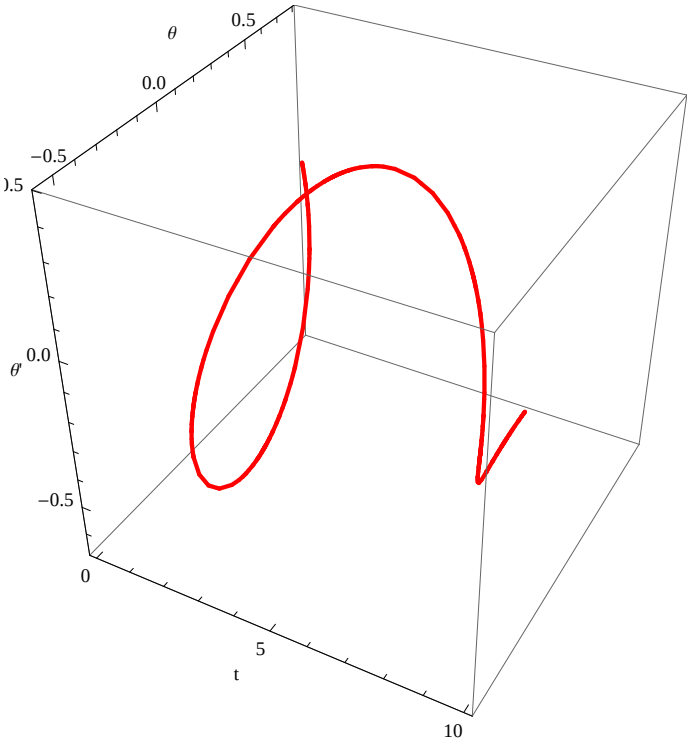
The phase portrait is also easy to create.

```
ParametricPlot[{numsoln[[1, 1, 2]], dnumsoln}, {t, 0, 10}, PlotStyle → Thick]
```



A three-dimensional view can sometimes be quite illustrative.

```
ParametricPlot3D[{t, numsoln[[1, 1, 2]], dnumsoln}, {t, 0, 10},
 BoxRatios → {1, 1, 1}, AxesLabel → {"t", "θ", "θ'"}, PlotStyle → {Red, Thick}]
```



You can now rotate the image by clicking in the plot and using your mouse to navigate.

To illustrate the use of numeric **NDSolve** for a system, consider the second-order system

```
ivp = {D[x[t], t] == 3 * x[t]^2 * y[t] - 6 * y[t]^2,
  D[y[t], t] == -x[t]^3 + 2 * x[t]^2 + 2 * x[t] * y[t] - 4 * y[t], x[0] == 1, y[0] == 0}
```

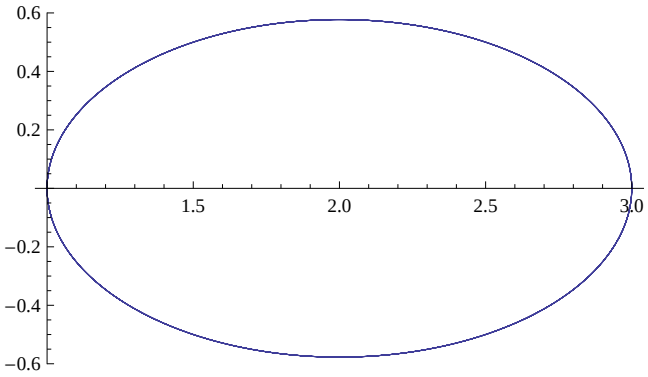
$$\{x'[t] == 3\,x[t]^2\,y[t] - 6\,y[t]^2,\ y'[t] == 2\,x[t]^2 - x[t]^3 - 4\,y[t] + 2\,x[t]\,y[t],\ x[0] == 1,\ y[0] == 0\}$$

The level curves of the Liapunov function $L(x, y) = (x - 2)^2 + 3y^2$are trajectories for this system. That is, = () L ,() x *t* y() *t* () L ,*x* 0 *y* 0 for all *t* > 0. This can be illustrated using the above methods:

```
numsoln = NDSolve[ivp, {x[t], y[t]}, {t, 0, 10}]
```

```
{{x[t] → InterpolatingFunction[{{0., 10.}}, <>][t],
  y[t] → InterpolatingFunction[{{0., 10.}}, <>][t]}}
```

```
ParametricPlot[{numsoln[[1, 1, 2]], numsoln[[1, 2, 2]]}, {t, 0, 10}]
```



■ Series Solutions (Chapter 8 of the Nagle/Saff/Snider text)

Some differential equations, and systems, are most approachable using power series techniques. *Mathematica* provides access to this method by way of the **Series** command.

Consider the initial value problem

$$\frac{d^2y}{dx^2} + \frac{1}{x}\frac{dy}{dx} - \frac{y}{x^2+1} = 0,\ y(-1) = c_1,\ y'(-1) = c_2$$

We begin by finding the series solution for $c_1 = 1, c_2 = 0$.

```
eqn = y''[x] + (1 / x) y'[x] - y[x] / (x^2 + 1) == 0;
lhs = Series[eqn[[1]], {x, -1, 5}];
coeffs = Solve[{lhs == 0, y[-1] == 1, y'[-1] == 0}];
ssoln1 = Series[y[x], {x, -1, 5}] /. coeffs
```

$$\left\{1 + \frac{1}{4}(x+1)^2 + \frac{1}{6}(x+1)^3 + \frac{11}{96}(x+1)^4 + \frac{1}{12}(x+1)^5 + 0[x+1]^6\right\}$$

To actually see the two linearly independent solutions we could now let $c_1 = 0$,and $c_2 = 1$ and solve again

```
eqn = y''[x] + (1 / x) y'[x] - y[x] / (x^2 + 1) == 0;
lhs = Series[eqn[[1]], {x, -1, 5}];
coeffs = Solve[{lhs == 0, y[-1] == 0, y'[-1] == 1}];
ssoln2 = Series[y[x], {x, -1, 5}] /. coeffs
```

$$\left\{(x+1)+\frac{1}{2}(x+1)^2+\frac{5}{12}(x+1)^3+\frac{1}{3}(x+1)^4+\frac{127}{480}(x+1)^5+0[x+1]^6\right\}$$

The beginning terms of two linearly independent solutions are evident. The solution to the initial value problem may thus be written as

$$c_1\left(1+\frac{1}{4}(t+1)^2+\frac{1}{6}(t+1)^3+\frac{11}{96}(t+1)^4+\frac{1}{12}(t+1)^5+O(t+1)^6\right)+$$
$$c_2\left((t+1)+\frac{1}{2}(t+1)^2+\frac{5}{12}(t+1)^3+\frac{1}{3}(t+1)^4+\frac{127}{480}(t+1)^5+O(t+1)^6\right)$$

To obtain additional terms in the series solution, we can modify the order requested in the **Series** command.

Mathematica will separately collect the terms involving c1 and c2.

```
c1 * ssoln1 + c2 * ssoln2
```

$$\left\{c1+c2\,(x+1)+\left(\frac{c1}{4}+\frac{c2}{2}\right)(x+1)^2+\left(\frac{c1}{6}+\frac{5\,c2}{12}\right)(x+1)^3+\right.$$
$$\left.\left(\frac{11\,c1}{96}+\frac{c2}{3}\right)(x+1)^4+\left(\frac{c1}{12}+\frac{127\,c2}{480}\right)(x+1)^5+0[x+1]^6\right\}$$

If we wish to plot or do other operations with the solution function, the solution must be converted to a polynomial.

```
Normal[c1 * ssoln1 + c2 * ssoln2][[1]]
```

$$c1+c2\,(1+x)+\left(\frac{c1}{4}+\frac{c2}{2}\right)(1+x)^2+\left(\frac{c1}{6}+\frac{5\,c2}{12}\right)(1+x)^3+\left(\frac{11\,c1}{96}+\frac{c2}{3}\right)(1+x)^4+\left(\frac{c1}{12}+\frac{127\,c2}{480}\right)(1+x)^5$$

For $c_1 = 1$ and $c_2 = 0$, we get

```
Normal[ssoln1][[1]]
```

$$1+\frac{1}{4}(1+x)^2+\frac{1}{6}(1+x)^3+\frac{11}{96}(1+x)^4+\frac{1}{12}(1+x)^5$$

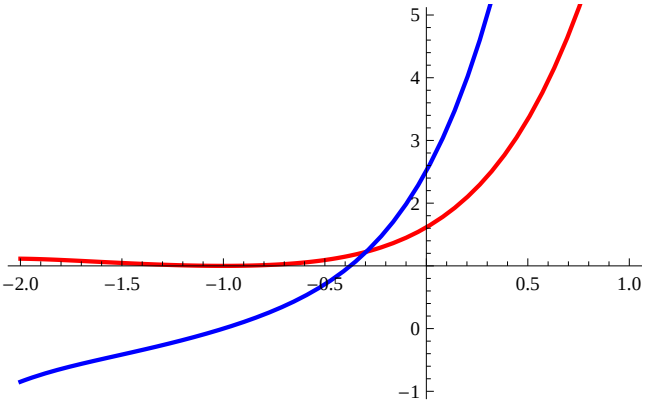
while for $c_1 = 0$ and $c_2 = 1$, we get

```
Normal[ssoln2][[1]]
```

$$1+x+\frac{1}{2}(1+x)^2+\frac{5}{12}(1+x)^3+\frac{1}{3}(1+x)^4+\frac{127}{480}(1+x)^5$$

These solutions may be plotted as follows:

```
g1 = Plot[Evaluate[Normal[ssoln1][[1]]], {x, -2, 1}, PlotStyle → {Red, Thick}];
g2 = Plot[Evaluate[Normal[ssoln2][[1]]], {x, -2, 1}, PlotStyle → {Blue, Thick}];
Show[g1, g2, PlotRange → {-1, 5}]
```



■ Laplace Transforms (Chapter 7 of the Nagle/Saff/Snider text)

The two commands that form the Laplace transform pair are **LaplaceTransform** and **InverseLaplaceTransform**.These commands can be used explicitly or implicitly to solve a differential equation. To illustrate both approaches, consider the problem of determining the general solution to

```
eqn = D[y[t], {t, 2}] + 3 * D[y[t], t] - 4 * y[t] == 0
```

$$-4\,y[t]+3\,y'[t]+y''[t]==0$$

One approach is to begin by applying the Laplace transform to this equation and solving for the Laplace transform, $Y(s)$, of the solution $y(t)$.

```
lapeqn = LaplaceTransform[eqn, t, s];
solnlap = Solve[lapeqn, LaplaceTransform[y[t], t, s]][[1, 1, 2]]
```

$$\frac{3\,y[0]+s\,y[0]+y'[0]}{-4+3\,s+s^2}$$

Next, the inverse Laplace transform is applied to find the solution $y(t)$.

```
InverseLaplaceTransform[solnlap, s, t]
```

$$\frac{1}{5}e^{-4\,t}\,(y[0]-y'[0])+\frac{1}{5}e^{t}\,(4\,y[0]+y'[0])$$

Compare this solution to that produced by **DSolve**.

```
DSolve[eqn, y[t], t]
```

$$\{\{y[t]\to e^{-4\,t}\,C[1]+e^{t}\,C[2]\}\}$$

For additional assistance, consult the Help menu on **LaplaceTransform** and **InverseLaplaceTransform**.

4. *Mathematica* Notebooks

Introduction

In the previous chapter we gave a general introduction to *Mathematica* and a more specific introduction on using *Mathematica* in calculus and differential equations. In this chapter, we present fourteen labs for use in an introductory class of differential equations. The idea of these labs is to coordinate the use of *Mathematica* with class instruction in differential equations.

The *Mathematica* notebooks deal with most of the important topics that are covered in an introductory course in differential equations. The labs are connected with the material that is presented in the text *Fundamentals of Differential Equations* by Nagle/Saff/Snider as well as other texts. The labs are not exhaustive in covering all of the topics and surely not all applications, but they do cover many of the topics and many applications.

The idea of the labs is to get the student to take an active part in learning differential equations. Among the benefits of using *Mathematica* in the teaching of differential equations are the following:

- the student will easily see in easy fashion the behavior of solutions by means of direction fields and phase planes;
- the student will come to understand the role that parameters play in differential equations;
- the student will be able to examine “what-if” situations to discover properties of differential equations;
- the student will be able to handle computationally intractable problems and generate numeric solutions; and
- the student will be able to solve various applications of differential equations that involve extensive computations.

The computer laboratory exercises that follow require the students to do some work and therefore take an active part in their learning of differential equations. Each lab presents explanatory material and some solved problems but each also includes problems for students to do. These problems range from simple problems, where the student must simply fill in blanks in the *Mathematica* code that is given, to more involved problems that require the students to supply some code on their own. In either case, the students must understand what is going on to supply the correct code. Additionally, the students are asked to summarize what they have observed and learned while doing the exercises.

The labs should be used with discretion depending on the class situation. Some labs deal with numeric techniques or graphical techniques, some labs deal more with theoretical issues, and others deal more with applications. There is much more material available here than can be used in a given semester. Instructors may wish to use only parts of some labs. Some of the labs are lengthier than others and some parts may need to be omitted.

What follows is a brief synopsis of each lab. **The parenthetical references after the title of each lab refer to appropriate portions of the Nagle/Saff/Snider text**.

- **Laboratory One (First-Order Differential Equations: Graphical Analysis Using Direction Fields) (Section 1.3)**

Using the **`VectorPlot`**, **`NDSolve`**, and **`Plot`** commands of *Mathematica*, the students are shown how to plot direction fields for first-order differential equations. First-order equations are examined from a graphical point of view. This rather straightforward lab requests students to examine the direction fields and make some observations about the behavior of solutions.

- **Laboratory Two (Numerical Solutions by Euler's Method) (Sections 1.4 and 3.6)**

In this laboratory, Euler's Method and an Improved Euler's Method are briefly explained. A computer implementation for each method is then given. Exercises are given to compare numerical solutions with the exact solution and to compare solutions given by the two methods. Students are also given an exercise to show that in some situations numerical techniques must be used when an analytical solution is not available.

- **Laboratory Three (First-Order Differential Equations - Analyzing Analytic Solutions) (Section 1.2 and Chapter 2)**

Using the **`DSolve`** command, students are shown how to let *Mathematica* generate general solutions to first-order equations and unique solutions to initial-value problems. Through exercises the students come to see how solutions depend on values of certain parameters. An application is given using a model for an alligator population. Using the **`VectorPlot`** command students also look at graphs of solutions and gain an understanding of the difference between autonomous and non-autonomous equations, and the students learn about equilibrium solutions of autonomous equations.

- **Laboratory Four (Picard's Method) (Project B of Chapter 1)**

This laboratory is an implementation of Project B in the first chapter of the text by Nagle/Saff/Snider. It deals with Picard's Method which is the basis for the existence and uniqueness proof for first-order initial-value problems. First, Picard's Method is explained as a procedure in which a differential equation is converted into an integral equation from which a sequence of functions is generated; hopefully this sequence converges to a solution of the differential equation. A *Mathematica* implementation of this procedure is given. Then students are given a series of exercises, involving Picard's method, that deal with initial value problems that have unique solutions. The students are then led through an exercise for an initial-value problem without a unique solution, and they are shown how Picard's method can lead to different results depending on the starting function. The objective here is to help students see the importance of the assumptions given in existence and uniqueness theorems.

- **Laboratory Five (Applications of First-Order Differential Equations) (Chapter 3)**

This is the first of two labs on applications of first-order differential equations. The idea is to model real world phenomena by means of differential equations. First, three models are discussed for population growth - the exponential model, the logistic model, and the modified logistic model. Direction fields are generated to study the behavior of solutions to differential equations modeling the growth. Then other phenomena are modeled by differential equations. Examples are (1) cooling modeled by Newton's Law of Cooling, (2) annuities (the differential equation model is compared to the difference equation model), and (3) the spread of diseases. The lab ends with a mixing problem.

- **Laboratory Six (Further Applications of First-Order Differential Equations) (Section 3.3, Project D of Chapter 2, Project D of Chapter 3)**

This is the second of two labs on applications of differential equations. There are basically three applications: the Snowplow Problem, Aircraft Guidance in a Crosswind, and the Heating and Cooling of Buildings. All of these applications are taken from the text by Nagle/Saff/Snider. *Mathematica* is a particularly good venue to use in solving these problems because parameters can be varied, "what-if" questions can be answered, and graphs of solutions can be generated. The last exercise on the heating and cooling of buildings is the most involved of the three and would be difficult without the computer to generate the solutions and the graphs. Most of the code to do this exercise is supplied.

Laboratory Seven (Higher-Order Differential Equations) (Chapters 4 and 6)

This lab begins the study of second-order and higher-order differential equations. The student is taught how to use built-in *Mathematica* commands to solve both homogeneous and non-homogeneous equations. For equations for which analytic solutions are not available, the student is shown how to generate numeric solutions by means of built-in *Mathematica* commands that give fourth-fifth order Runge-Kutta solutions. The lab then continues by looking at other ways *Mathematica* can solve both homogeneous and non-homogeneous equations. These ways are (1) finding a solution to homogeneous equations by the traditional method of solving the auxiliary equation and then finding a fundamental solution set and (2) using built-in *Mathematica* commands to find a basis for the set of solutions to the homogeneous equation. Then the lab shows how to use built-in *Mathematica* commands to solve non-homogeneous equations by the variation of parameters method. A nice application regarding transverse vibrations of a beam is given in the lab.

- **Laboratory Eight (An Application of 2^{nd} Order Equations – The Mass-Spring System) (Sections 4.1-4.3 and 4.8)**

In this lab harmonic oscillation is studied by means of mass-spring systems. The mass-spring system is considered first without damping and then with damping. Physical interpretations of the mass-spring from the graphical representations of the solutions of the differential equation models are an important ingredient of this lab. In the damped cases, students can use *Mathematica* to vary the damping coefficient in the models and see the effect upon the behavior of the solution. In this way the critical damping value becomes real for the students. In the latter part of the lab, students are led through an exercise to determine the time value when the harmonic oscillator essentially comes to rest.

- **Laboratory Nine (The Pendulum) (Section 4.8)**

As another application of second-order differential equations, the motion of a pendulum is modeled by a non-linear second-order differential equation in this lab. Being nonlinear, this equation is an interesting object of study. *Mathematica* can handle it nicely with numeric techniques. First, the undamped case is considered, and the behavior of the pendulum is studied for various changes in initial conditions. Students are required to make inferences on the pendulum's behavior from the graphs of solutions. Next, the linearized version of the undamped case is considered, and students should see when it is appropriate to linearize the model. Finally, the damped case is considered both for the original model and the linearized version. The effect of the damping coefficient is studied by means of the built-in *Mathematica* commands. The lab ends with an interesting example of a pendulum increasing in length, called the Poe Pendulum.

- **Laboratory Ten (Forced Equations and Resonance) (Section 4.10)**

In this lab we take a deeper look at second-order, non-homogeneous equations. We concentrate on equations that have a periodic forcing term so that we could think of this as modeling a mass-spring system with an external periodic stimulus. First, we take the undamped case and look at the parts of the solutions generated by *Mathematica* that comprise the particular solution due to the forcing term. We examine the effect of changing the frequency of the forcing term and discuss the frequency gain function. Next the lab continues by considering the damped case and has exercises to determine the steady-state parts of solutions and transient parts of solutions. We again examine the effect of changing the frequency of the forcing term and look at the graph of the frequency gain function. This examination leads to the discussion of the resonance frequency and what it means for a system to be at resonance.

Laboratory Eleven (Laplace Transformations and Projectile Motion) (Chapter 7 and Section 3.4)

The focus of this lab is to understand the Laplace transform and the inverse Laplace transform and then to learn to use the transforms in the solutions of differential equations. First, the students are shown how to compute the transform much as they would by pencil and paper but instead by using *Mathematica* to do the calculations; they are then shown how to let *Mathematica* compute the transform outright. Next, the students are shown how to solve initial-value problems by using Laplace transforms much as they would using pencil and paper but instead by using *Mathematica* to find the transforms. Then the students are shown how to solve initial-value problems using the Laplace transform commands in *Mathematica*. The lab concludes with instructions on using Laplace transforms to solve systems of differential equations, and the lab presents applications of this procedure to the study of projectile motion.

- **Laboratory Twelve (Analytical and Graphical Analysis of Systems) (Sections 5.1, 5.2 and 5.4 and Sections 9.1 – 9.6)**

The focus of this lab is two-dimensional linear systems, but much of the information given could also be used for higher-dimensional systems. The lab utilizes *Mathematica's* linear algebra capabilities to compute eigenvalues and eigenvectors of the systems. With those computations at hand, the student can construct the solution of the system. Then using the `VectorPlot` command, the phase plane is generated, and students can see the behavior of solutions for the various cases that can arise: two distinct positive real eigenvalues, two real eigenvalues of opposite sign, only one real eigenvalue, and so forth. Next, complex eigenvalues are discussed, and the general solution for the system is generated for this situation. Finally, there are some exercises dealing with second-order equations to be solved as systems and three-dimensional systems, and students are shown how to solve systems by *Mathematica* without first finding the eigenvalues and eigenvectors.

- **Laboratory Thirteen (Numerical Techniques on Systems) (Sections 3.6, 3.7, and 5.3)**

In this lab we consider methods of solving first-order systems by numerical techniques. We consider linear and nonlinear systems as well as autonomous and non-autonomous systems. Various numerical techniques are presented and compared. These techniques include Euler's Method, a variation of Euler's Method, an improved Euler's Method, and the Runge-Kutta Method. Exercises are given to employ these methods on various systems and second-order equations. The graphs and phase plane of solutions are generated to aid in the discussion about the behavior of solutions.

- **Laboratory Fourteen (Series Solutions) (Sections 8.1 – 8.4)**

In this lab we deal with finding series solutions to second-order linear equations. Students are shown *Mathematica* commands that can be used to find series solutions about ordinary points. Then graphs and tables of values of solutions are given so that the convergence of the series solutions can be analyzed. Among the exercises that are given are exercises dealing with Legendre polynomials and aging springs.

Laboratory One

First Order Differential Equations

Graphical Analysis Using Direction Fields

(Section 1.3 of the Nagle/Saff/Snider text)

In this section we will look at first-order differential equations from a graphical point of view and we will make use of direction fields. Our objectives are as follows.

1. To learn how to use the computer to create the direction field for the differential equation.
2. To learn how the direction field gives pertinent information about solutions to differential equations.
3. To see how actual solutions fit onto the direction field.

New commands to be used in this section are the **`VectorPlot`**, **`NDSolve`**, **`Plot`**, and **`Show`** commands.

1.1 Direction Fields

A *direction field* for a differential equation is a two-dimensional plot comprised of tiny line segments that are portions of the tangent lines to solutions of the differential equation. Without knowing solutions, the direction field gives some valuable qualitative information about solutions. A method known as the *method of isoclines* is a means of constructing the direction field. The construction is accomplished by setting the slope equal to various constants c and plotting the level curve $f(t, y(t)) = c$ for those values for c. For each c, the slope of the solution $\boldsymbol{x}(t)$ is the same.

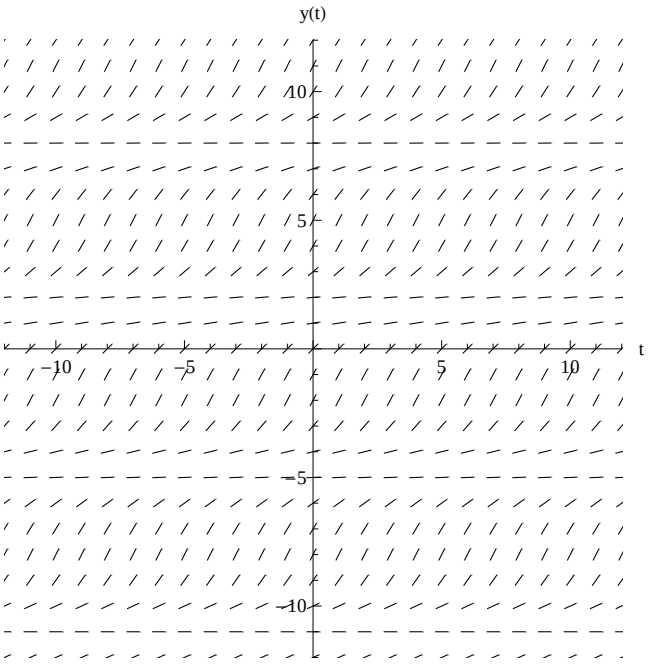
Because explicit solutions to a differential equation cannot always be derived, direction fields are valuable in studying the behavior of solutions. Yet, even by using the method of isoclines finding the direction field can be very labor intensive. Fortunately, in *Mathematica* a direction field can be constructed using the **`VectorPlot`** command.

Let's start with an example, the differential equation

$$\frac{d\,y(t)}{d\,t} = 1 - \sin(y(t))$$

In order to create the direction field for this equation we will first use the **`VectorPlot`** command.

```
dir = VectorPlot[{1, 1 - Sin[y]}, {t, -12, 12}, {y, -12, 12}, VectorPoints → 25, Axes → True,
  Frame → False, VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.015, 1, None},
  PlotRange → {{-12, 12}, {-12, 12}}, AxesOrigin → {0, 0}, AxesLabel → {"t", "y(t)"}]
```



1. Answer the following questions regarding what you see.

(a) If a solution has the initial value of 4 at 0, that is, $y(0) = 4$, what does it appear will be the behavior of the solution as t gets larger? As t gets smaller? Between what values will the solution $y(t)$ always be for any choice of t?

(b) What do you notice about the direction of the little vectors for any given value of y? From the differential equation (particularly the right hand side of it), is there good rationale for your answer? What is it?

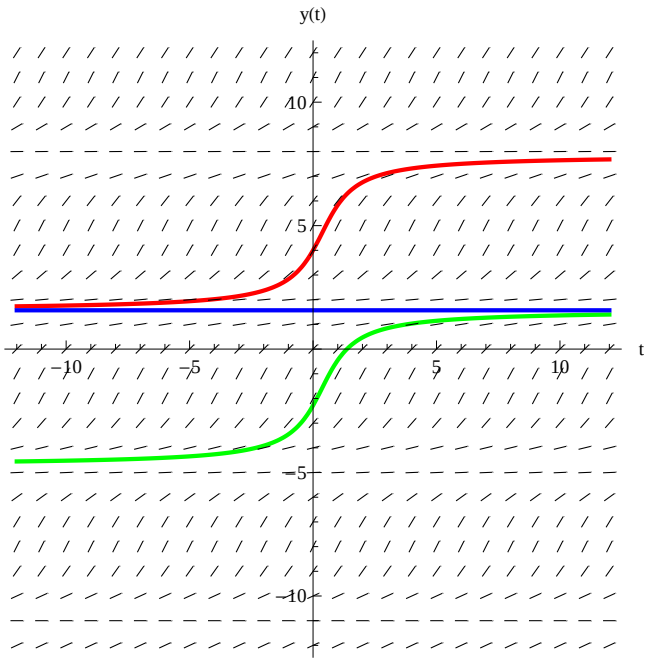
(c) If a solution had an initial value of $y(0) = 4 - 2\pi$, what would be the general shape of the solution in terms of the solution with initial value of $y(0) = 4$,? What is the reason for your answer when you consider the differential equation itself?

(d) What are equilibrium solutions, that is, solutions that are constant?

Let's now check by actually plotting some of the solutions to see if your observations were correct. To do this we will enter solve the equation using the **NDSolve** command and plot them over the direction field using the **Plot** and **Show** commands.

```
Clear[x, y]
```

```
soln1 = NDSolve[{y'[t] == 1 - Sin[y[t]], y[0] == 4}, y, {t, -12, 12}];
g1 = Plot[y[t] /. soln1, {t, -12, 12}, PlotStyle → {Thick, Red}];
soln2 = NDSolve[{y'[t] == 1 - Sin[y[t]], y[0] == 4 - 2 π}, y, {t, -12, 12}];
g2 = Plot[y[t] /. soln2, {t, -12, 12}, PlotStyle → {Thick, Green}];
soln3 = NDSolve[{y'[t] == 1 - Sin[y[t]], y[0] == π/2}, y, {t, -12, 12}];
g3 = Plot[y[t] /. soln3, {t, -12, 12}, PlotStyle → {Thick, Blue}];
Show[g1, g2, g3, dir, AspectRatio → 1, AxesLabel → {"t", "y(t)"},
 AxesOrigin → {0, 0}, PlotRange → {{-12, 12}, {-12, 12}}]
```



Do you need to change any of your previous answers? Before proceeding with the next exercise, it may be wise to use the **Clear** command to reinitialize all variables or at least to reinitialize individual variables.

2. Consider the differential equation

$$\frac{dy}{dt} = y(y-2)(y+1)$$

(a) Examine the direction field using the commands above and sketch various representative solutions in the window $0 < t < 5$ and $-2 < y < 4$.

(b) If $y(0) = 1.9$, what is the limit of $y(t)$ as t goes to infinity?

(c) If $y(0) = 2.1$, what is the limit of $y(t)$ as t goes to infinity?

(d) Do any of the solution curves ever cross each other? Do you know why? (See the Theorem 1 in Section 1.2 of the text by Nagle, Saff, and Snider.)

(e) If y is initially .5 (i.e. $y(0) = .5$), can y eventually be -1? Why?

3. Using the commands above, determine the direction field and some solution curves for each of the following differential equations in the window $-5 < t < 5$ and $-5 < y < 5$.

(a) $\frac{dy}{dt} = \sin(t)$. Add various representative solutions curves. Are there any equilibrium solutions? Characterize the appearance of the solutions.

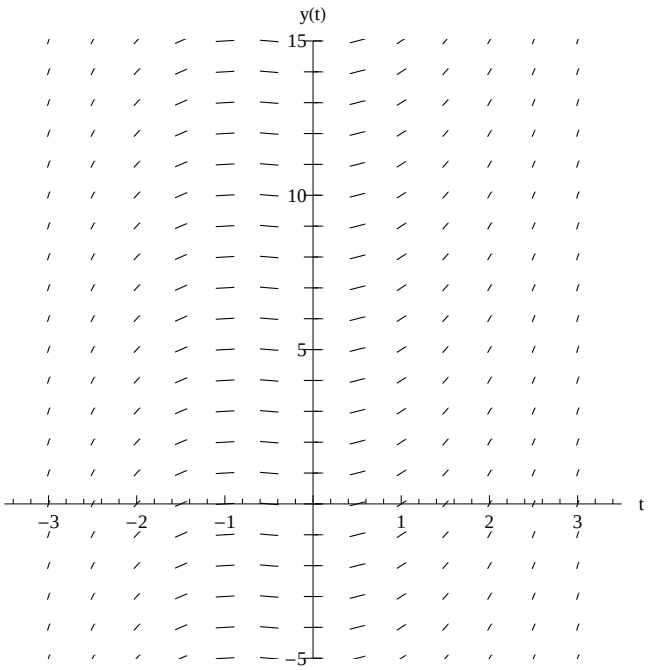
(b) $\frac{dy}{dt} = \sin(y)$. Add various representative solution curves. Are there any equilibrium solutions? Characterize the appearance of the solutions.

(c) $\frac{dy}{dt} = \sin(y)\sin(t)$. Add various representative solution curves. Are there equilibrium solutions? Characterize the appearance of the solutions.

1.2 Using VectorFieldPlot, NDSolve, Plot, and Show

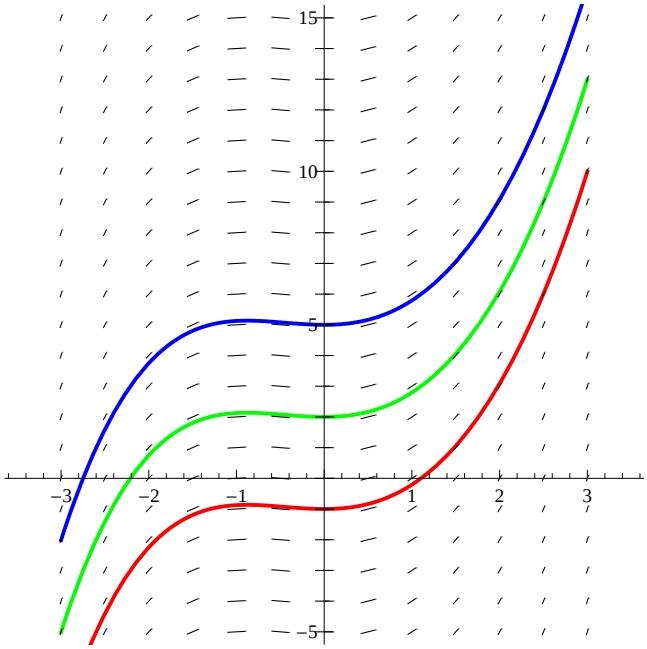
Next we will use the **VectorPlot**, **NDSolve**, **Plot**, and **Show** commands to look at direction fields. It is probably a good idea to use the **Clear** command before we proceed.

```
Clear[y, g1, g2, g3, dir];
dir = VectorPlot[{1, t^2 + Sin[t]}, {t, -3, 3}, {y, -5, 15},
  PlotRange → {{-3.5, 3.5}, {-5, 15}}, VectorPoints → {13, 21}, Axes → True,
  Frame → False, VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.01, 1, None},
  VectorStyle → {Black, Arrowheads[Medium]}, AspectRatio → 1, AxesLabel → {"t", "y(t)"}]
```



A family of solutions can be plotted within the direction field by the **NDSolve**, **Plot**, and **Show** commands. All we need to do is specify some initial conditions.

```
soln1 = NDSolve[{x'[t] == t^2 + Sin[t], x[0] == -1}, x, {t, -3, 3}];
g1 = Plot[x[t] /. soln1, {t, -3, 3}, PlotStyle → {Thickness[0.006], Red}];
soln2 = NDSolve[{x'[t] == t^2 + Sin[t], x[0] == 2}, x, {t, -3, 3}];
g2 = Plot[x[t] /. soln2, {t, -3, 3}, PlotStyle → {Thickness[0.006], Green}];
soln3 = NDSolve[{x'[t] == t^2 + Sin[t], x[0] == 5}, x, {t, -3, 3}];
g3 = Plot[x[t] /. soln3, {t, -3, 3}, PlotStyle → {Thickness[0.006], Blue}];
Show[g1, g2, g3, dir, AspectRatio → 1, PlotRange → {{-3.5, 3.5}, {-5, 15}}]
```



1.3 More Exercises

4. By examining the direction fields, what can one say about the behavior as t approaches ∞ of solutions to

$$\frac{dx}{dt} = 2 - x + \frac{1}{t}?$$

Do small variations in initial conditions lead to small or large variations in the solutions? Are the solutions periodic? If so, what is the period? Do the solutions tend to an asymptote? If so, what is that asymptote?

5. By examining the direction fields, what can one say about the behavior as t approaches ∞ of solutions to

$$\frac{dy(t)}{dt} = \sin(t) - \frac{x}{2}?$$

Do small variations in initial conditions lead to small or large variations in the solutions? Are the solutions periodic? If so, what is the period? Do the solutions tend to an asymptote? If so, what is that asymptote?

6. For what values is the differential equation

$$\frac{dx}{dt} = -\frac{t}{x}$$

defined? Illustrate and discuss the behavior of solutions to this differential equation.

7. Find some functions $f(t, x)$ for which the solutions of

$$\frac{dx}{dt} = f(t, x)$$

are symmetrical with respect to the x-axis. Experiment with creative choices for $f(t, x)$ and then make some concluding conjecture. Test your conjecture.

Laboratory Two

Numerical Solution by Euler's Method

(Sections 1.4 and 3.6 of the Nagle/Saff/Snider text)

Applications of differential equations to problems, which have their basis in questions that arise from real world situations, frequently have no readily available closed form solutions. In these cases, approximations to the solutions are the best we can hope for. The simplest way to approximate a solution is to piece together the field lines of the differential equation. Our objectives in this section are as follows.

1. To learn how to numerically approximate solutions by means of Euler's method and the improved Euler's method.

2. To see how the methods compare with each other and the actual solution if it is available.

3. To see how numerical schemes like Euler's method are valuable when actual analytic solutions are not readily available.

Key *Mathematica* commands in this lab are the **`ListPlot`** and **`Show`** commands.

2.1 Euler's Method

Consider a differential equation of the form $\frac{dx}{dt} = f(t, x),\ x(t_0) = x_0$. To employ Euler's method, named after the prolific 18th century Swiss mathematician Euler who developed it, we begin by calculating the slope of the solution at the initial position, and then we use this slope to approximate the solution by a straight line. From here, we increase t by an incremental fixed amount, called the step size, and calculate the slope of the solution through the resulting point on the line, and then we repeat the process.

To be precise, let us recall that the line segment joining two points $(t_n,\ x_n)$ and $(t_{n+1},\ x_{n+1})$ on a curve is called a *chord* or *secant line*. The slope of the chord may be represented by

$$\text{slope} = \frac{x_{n+1} - x_n}{t_{n+1} - t_n}.$$

If $(t_n,\ x_n)$is a point on an unknown curve $x(t)$, then the derivative $\frac{dx}{dt}$at the point $(t_n,\ x_n)$ can be used to approximate the coordinate $(t_{n+1},\ x_{n+1})$. Setting the derivative equal to its approximate value, that is

$$\frac{x_{n+1} - x_n}{t_{n+1} - t_n} = \frac{dx}{dt}$$

provided t_{n+1} is close to t_n, and then clearing the fractions yields $x_{n+1} - x_n = (t_{n+1} - t_n)\frac{dx}{dt}$, or equivalently,

$$x_{n+1} = x_n + h\frac{dx}{dt}$$

where $h = t_{n+1} - t_n$. If $\frac{dx}{dt} = f(t, x)$, then $x_{n+1} = x_n + h f(x_n, t_n)$. The use of this iteration formula together with the iteration

$$t_{k+1} = t_k + h,$$

where *h* is the step size, is used to approximate points on the curve of the unknown function $x(t)$ by Euler's method.

For example we can apply Euler's method to

$$\frac{dx}{dt} = x \text{ with initial condition } x(0) = 1.$$

To numerically solve via Euler's method, we begin by declaring our known initial conditions.

```
f[t_, x_] := x;
t[0] = 0;
x[0] = 1;
h = .1;
```

Unless the solution is a straight line, the point (t_1, x_1) does not lie on the curve but on the straight line tangent to the curve at the point (t_0, x_0). We approximate this as follows.

```
t[1] = t[0] + h
```

```
0.1
```

```
x[1] = x[0] + h * f[t[0], x[0]]
```

```
1.1
```

```
t[2] = t[1] + h
```

```
0.2
```

```
x[2] = x[1] + h * f[t[1], x[1]]
```

```
1.21
```

```
t[3] = t[2] + h
```

```
0.3
```

```
x[3] = x[2] + h * f[t[2], x[2]]
```

```
1.331
```

```
t[4] = t[3] + h
```

```
0.4
```

```
x[4] = x[3] + h * f[t[3], x[3]]
```

```
1.4641
```

```
t[5] = t[4] + h
```

```
0.5
```

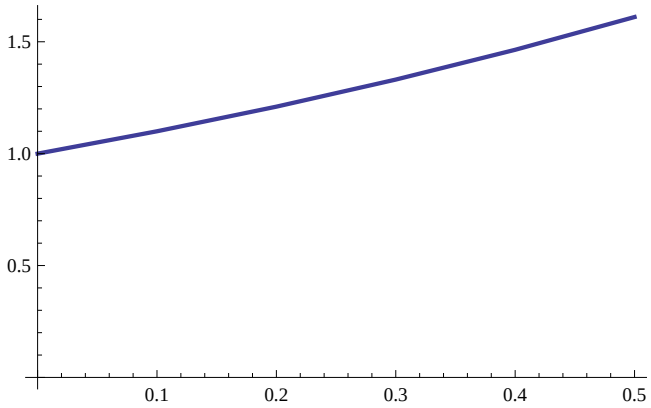
```
x[5] = x[4] + h * f[t[4], x[4]]
```

```
1.61051
```

```
ptlist = Table[{t[i], x[i]}, {i, 0, 5}]
```

```
{{0, 1}, {0.1, 1.1}, {0.2, 1.21}, {0.3, 1.331}, {0.4, 1.4641}, {0.5, 1.61051}}
```

```
ListPlot[ptlist, Joined → True, PlotStyle → Thick, AxesOrigin → {0, 0}]
```



We can refine this procedure into a formal procedure as shown here.

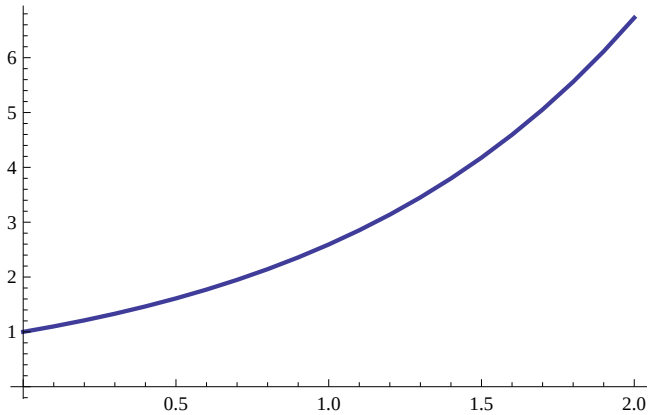
```
EulersMethod[t0_, x0_, h_, halt_] :=
  Module[{t = t0, x = x0, j, m = halt},
   T = Table[t + j * h, {j, 0, m}];
   X = Table[x, {j, 0, m}];
   For[j = 1, j ≤ m, j++,
    X[[j + 1]] = X[[j]] + h * f[T[[j]], X[[j]]]];
   Return[Transpose[{T, X}]]];
```

We invoke the **EulersMethod** procedure with an increment of .1 and halt after 20 steps as follows.

```
ptlist = EulersMethod[0, 1, .1, 20]
```

```
{{0, 1}, {0.1, 1.1}, {0.2, 1.21}, {0.3, 1.331}, {0.4, 1.4641}, {0.5, 1.61051},
 {0.6, 1.77156}, {0.7, 1.94872}, {0.8, 2.14359}, {0.9, 2.35795}, {1., 2.59374},
 {1.1, 2.85312}, {1.2, 3.13843}, {1.3, 3.45227}, {1.4, 3.7975}, {1.5, 4.17725},
 {1.6, 4.59497}, {1.7, 5.05447}, {1.8, 5.55992}, {1.9, 6.11591}, {2., 6.7275}}
```

```
ListPlot[ptlist, Joined → True, PlotStyle → Thick, AxesOrigin → {0, 0}]
```



The solution of this initial-value problem is the exponential function with respect to variable *t*.

2.2 Exercises

1. Consider the initial value problem $\frac{dx}{dt} = (3 - x)\,x$ where $x(0) = 2$. Compare Euler's method with the exact solution by taking h=.1 and n=20 iterations. Graph the real solution and the approximate solution together on one graph and make a table listing the values of the actual solution and the approximate solution for the values 0, .1,.2,...,2. (Hint: You may want to use the commands given below where you must supply the needed data at the spots marked **fill**.) Comment on the (in)accuracy of the approximation.

```
Clear[x, t, f];
f[t_, x_] := fill;
t0 = fill; x0 = fill;
EulersMethod[t0, x0, .1, fill];
ptlist = EulersMethod[t0, x0, .1, fill];
plot1 = ListPlot[ptlist, Joined -> True];
soln1 = NDSolve[{x'[t] == (3 - x[t]) * x[t], x[0] == fill}, x, {t, 0, 2}];
plot2 = Plot[x[t] /. soln1, {t, 0, 2}];
Show[plot1, plot2]
soln2 = NDSolve[{x'[t] == (3 - x[t]) * x[t], x[0] == fill}, x, {t, 0, 2}];
Table[Flatten[{ptlist[[i]][[1]], ptlist[[i]][[2]], x[ptlist[[i]][[1]]] /.
soln2}], {i, 1, fill}]
```

2. (a) Use the **DSolve** command to try to solve $\frac{dx}{dt} = e^{-t^2}$. What can you say about *Mathematica*'s capability to solve differential equations in terms of elementary functions based on this result?

(b) Use Euler's method to numerically approximate the solution to (a) on the interval [1,2] with the initial value x(1) =1.

2.3 Improved Euler's Method

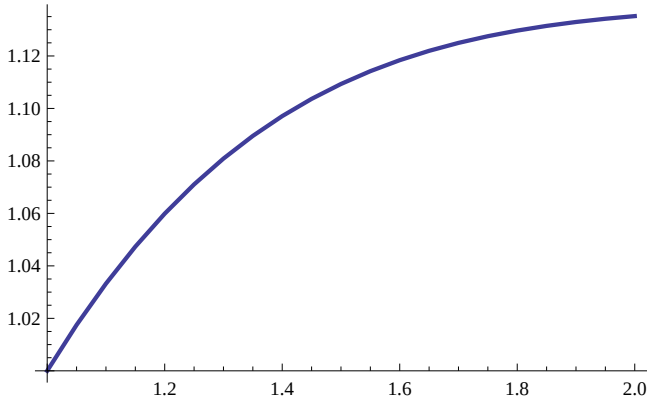
We can improve upon Euler's method by replacing the slope by the average of two slopes. The equations become

$$t_{n+1} = t_n + h$$
$$x_{n+1} = x_n + (h/2)\,(f(t_n, x_n) + f(t_{n+1}, x_{n+1})) \quad \text{which by Euler's approximation becomes}$$
$$x_{n+1} = x_n + (h/2)\,(f(t_n, x_n) + f(t_n + h, x_n + h\,f(t_n, x_n)))$$

Notice that we can compute successive values of *t* and *x* from preceding values. We can incorporate this computation into a procedure as follows.

```
Clear[t, x, f, t0, x0, ptlist];
f[t_, x_] := Exp[-t^2];
ImprovedEuler[t0_, x0_, h_, halt_] :=
  Module[{t = t0, x = x0, j, m = halt},
   T = Table[t + j * h, {j, 0, m}];
   X = Table[x, {j, 0, m}];
   For[j = 1, j ≤ m, j++,
    X[[j + 1]] = X[[j]] + h * f[T[[j]] + (h / 2), X[[j]] + (h / 2) * f[T[[j]], X[[j]]]]];
   Return[Transpose[{T, X}]]];
```

```
ptlist = ImprovedEuler[1, 1, 0.05, 20];
ListPlot[ptlist, Joined → True, PlotStyle → Thick, AxesOrigin → {1, 1}]
```



2.4 Exercises

3. By the improved Euler's method, solve the initial value problem $\frac{dx}{dt} = 5x - 6e^{-t}$, $x(0) = 1$ on the interval [0,1]. Compare the values at t = 1 when h is taken to be .1, then .05, and finally .025. Compare these results with the actual value of x(1).

4. A child situated at the point (0, y_0) on the positive y axis is dragging a toy on a string, which is "a" units long. She begins by grasping one end of the taut string when the toy connected to the other end rests at the point (x_0, 0) on the positive x axis. She walks at a constant pace along the positive y axis, away from the origin.

(a) Show that the differential equation which describes the slope of the tangent line to the curve traced out by the toy is $\frac{dx}{dt} = \frac{\sqrt{a^2 - x^2}}{x}$, where y($x_0$) = 0. (Hint: Draw the graph of the anticipated path of the toy and examine the slope of the tangent line.)

(b) Use both Euler's method and improved Euler's method with a = 10, x_0= 8, h = -.1, and the number of iterations n = 75 to approximate the solution to this equation. Graph them both on the same axis (using the **Show** command) and compare the results. (Caution: When Euler's method or the improved Euler's method are used, make sure variables are renamed if necessary.)

(c) Let *Mathematica* find the actual solution and its graph. How does it compare with the graphs of the approximate solutions?

Laboratory Three

First-Order Differential Equations

Analyzing Analytic Solutions

(Section 1.2 and Chapter 2 of the Nagle/Saff/Snider text)

This lab continues the study of first-order differential equations and uses the computer to analyze solutions and determine some of their important properties. Our objectives are as follows.

1. To learn how to use *Mathematica* to generate when possible the actual solutions and families of solutions of first-order equations and determine properties of the solutions.

2. To determine the effect of parameters on differential equations.

3. To gain a better comprehension and appreciation for some of the theory of first-order differential equations.

Key *Mathematica* commands used in this section are the **`DSolve`** command and the **`VectorPlot`** command.

3.1 Solution Curves

A first-order ordinary differential equation is a statement about the slope of an unknown function and may be expressed as $\frac{dx}{dt} = f(t, x)$ or equivalently in *Mathematica* as **`x'[t]==f[t,x[t]]`** where variable *x*, since it is dependent upon variable *t*, must be input as **`x[t]`**. A solution is a differentiable function *x*(*t*) which satisfies the equation over some specified range of *t* values. The **`DSolve`** command can derive a general solution *x*(*t*) to most differential equations using the syntax

```
DSolve[x'[t]==f[t,x[t]],x[t],t]
```

Consider the following differential example.

$$\frac{dx}{dt} = t^2 + \sin(t)$$

Using *Mathematica* to solve for *x*(*t*) yields

```
DSolve[x'[t] == t^2 + Sin[t], x[t], t]
```

$$\left\{\left\{x[t] \to \frac{t^3}{3} + C[1] - \text{Cos}[t]\right\}\right\}$$

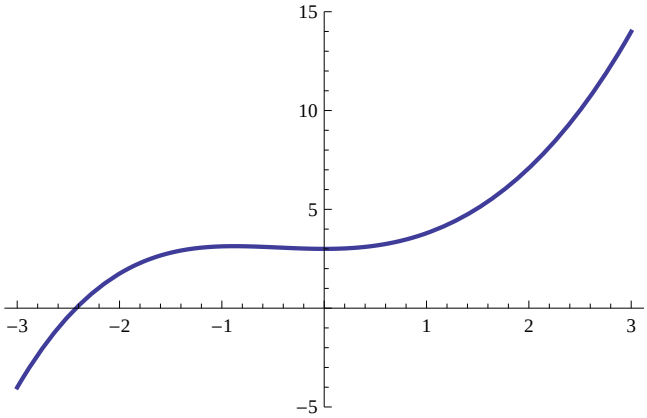
which is in terms of an arbitrary constant `C[1]`. Higher order equations yield several arbitrary constants. The solution *x*(*t*) can be completely determined by imposing some initial condition. If we assign an initial amount $x_0 = 3$ at time $t = 0$, then a specific solution to the above equation is

```
soln = DSolve[{x'[t] == t^2 + Sin[t], x[0] == 3}, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{1}{3}\left(12 + t^3 - 3\,\text{Cos}[t]\right)\right\}\right\}$$

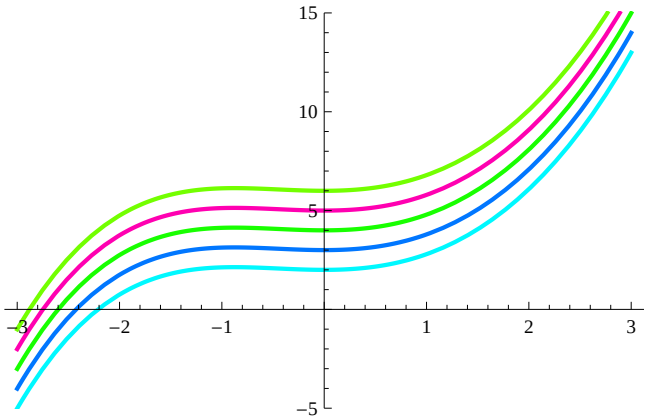
where the initial condition is enclosed with the original equation by braces {}. We can plot this specific solution to by executing the following commands:

```
Plot[x[t] /. soln, {t, -3, 3}, PlotRange → {-5, 15}, PlotStyle → Thick]
```



We can plot a family of solution curves with respect to $t_0 = 0$ by solving the initial value problem in terms of an unknown constant *c.* A plot of a family of solutions for the set {2, 3, 4, 5, 6} is shown below.

```
plotlist = {};
For[c = 2, c ≤ 6, c++,
  soln = DSolve[{x'[t] == t^2 + Sin[t], x[0] == c}, x[t], t]; AppendTo[plotlist,
   Plot[x[t] /. soln, {t, -3, 3}, PlotRange → {-5, 15}, PlotStyle → {Hue[Random[]], Thick}]];
Show[
 plotlist]
```



A plot of a family of solution curves is a means in determining how sensitive solution(s) of a differential equation are to the initial condition(s).

3.2 Exercises

1. Using the **DSolve** command, solve $\frac{dx}{dt} = kx$ and from the solution determine how many initial conditions are needed to completely determine a unique solution to $x' = k\,x$, where k is a constant. Explain. Similarly, using **DSolve**, determine how many initial conditions are needed to completely determine a unique solution to $x'' = k\,x$. Can you generalize your conclusions?

2. Using the **DSolve** command, what can you say about solutions to $x' = x$ that satisfy the initial conditions $x(0) = 1$ and $x(1) = e$?

3. Use the **DSolve** command of *Mathematica* to solve the differential equation $\frac{dy}{dx} = \frac{xy(x)}{x^2-1}$ for the initial conditions y(0) = c for various values of c. Describe the curves on which the solutions lie for each value of c. The following commands may be useful.

```
plotlist = {};
For[c = 1, c ≤ 6, c++,
  soln = Flatten[DSolve[{y'[x] == x y[x]/(x^2 - 1), y[0] == c}, y[x], x]]; AppendTo[plotlist,
   Plot[y[x] /. soln, {x, -1, 1}, PlotRange → {0, 10}, PlotStyle → {Hue[Random[]], Thick}]]];
Show[
 plotlist]
```

3.3 Existence and Uniqueness of Solutions

Consider a situation where quantity x changes with respect to time t at a rate equal to the cube root of the quantity that is currently present, that is

$$\frac{dx}{dt} = x^{1/3}$$

One existence and uniqueness theorem states that if $f(t, x)$ and $\frac{\partial}{\partial x} f(t, x)$ are continuous at every point within a rectangular region containing the initial point , (t_0, x_0), then a solution to the initial-value problem

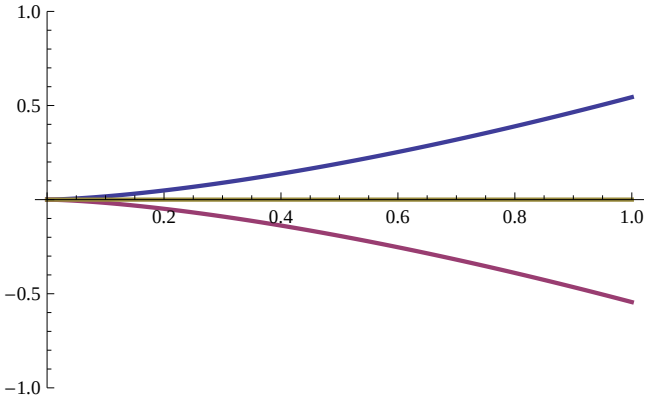
$$\frac{dx}{dt} = f(t, x) \text{ with } x(t_0) = x_0$$

exists on some small interval centered at $t_0$and is the only solution for that initial-value problem over that interval. If the initial assumptions are not satisfied, a solution to the initial-value problem may or may not exist and may or may not be unique. For the equation in Section 3.1above, both $f(t, x) = t^2 + \sin(t)$and $\frac{\partial}{\partial x} f(t, x) = 0$ are continuous functions for every point in a neighborhood of the initial value $x(0) = c$ where c is a constant. As a result, a solution to the initial-value problem always exists and is unique. For the above equation, $f(t, x) = x^{1/3}$is a continuous function but $\frac{\partial}{\partial x} f(t, x) = \frac{1}{3} x^{-2/3}$evaluated at the initial value $x(0) = 0$ is undefined and hence this partial derivative is discontinuous. The solution satisfying $x(0) = 0$ may not exist or may not be unique. Using **DSolve** to solve this initial- value problem yields one solution. Other solutions for this intial-value problem are the trivial solution $x(t) = 0$ and the negative of the solution obtained below.

```
soln = DSolve[{x'[t] == x[t]^(1 / 3), x[0] == 0}, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{2}{3}\sqrt{\frac{2}{3}}\, t^{3/2}\right\}\right\}$$

```
Plot[{x[t] /. soln, -x[t] /. soln, 0}, {t, 0, 1}, PlotRange → {-1, 1}, PlotStyle → Thick]
```



3.4 Exercises

4. Using *Mathematica* determine the solution of the initial value problem $\frac{dx}{dt} = x^{1/5}$ where $x(0) = 0$. How many solutions does *Mathematica* give? Are there more solutions?

5. Consider the first-order differential equation $\frac{dx}{dt} = -\frac{x}{1+t^2}$.

(a) Does this equation have a unique solution through (0 0)?

(b) If so, can you tell what it is? If you cannot, use *Mathematica* to explain.

(c) Does this equation have a unique solution through the point (0, -1)?

(d) Without graphing, give some important characteristics of the graphs of a solution through a point (0, *c*) for any *c* between -1 and 0 in relation to the graph of the solution through (0, -1).

6. For what values is the differential equation $\frac{dx}{dt} = -\frac{t}{x}$defined? Illustrate and discuss the behavior of solutions to this differential equation.

3.5 (Non)Autonomous Equations and Equilibria (Project C of Chapter 1 of the Nagle/Saff/Snider text)

A first-order ordinary differential equation is *autonomous* if it can be written in the form $\frac{dx}{dt} = f(x)$, where the slope is a function of the dependent variable *x* only and not on time. An equation which is not autonomous (called *non-autonomous*) is one in which $\frac{dx}{dt} = f(t, x)$, where f depends explicitly on *t*. We say that a solution *x(t)* of an autonomous equation is an *equilibrium solution* if $f(x) = 0$ and call the values *x* which satisfy $f(x) = 0$ the *equilibrium values.*

7. Experiment with the two differential equations $x' = x^2 + \sin(x)$and $x' = t^2 + \sin(x)$, one being autonomous and the other non-autonomous and demonstrate by appropriate graphs that the slopes of the direction field are in one case dependent only upon the *x* values while in the other case the slopes are also dependent upon the values of *t*. In which case is it true that the solution with initial condition x(0)=1 has the same appearance as the solution with the initial condition x(2)=1? Can you explain why? Make a general true statement about this. The following commands may be helpful; supply the data at the spots marked **fill**.

```
f[t_, x_] := fill;
dir = VectorPlot[{1, f[t, x]}, {t, fill, fill}, {x, fill, fill},
  VectorPoints → 15, Axes → True, Frame → False,
  VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.02, 1, None},
  PlotRange → {{fill, fill}, {fill, fill}}, AxesOrigin → {0, 0}, AxesLabel → {"t", "y(t)"}]
```

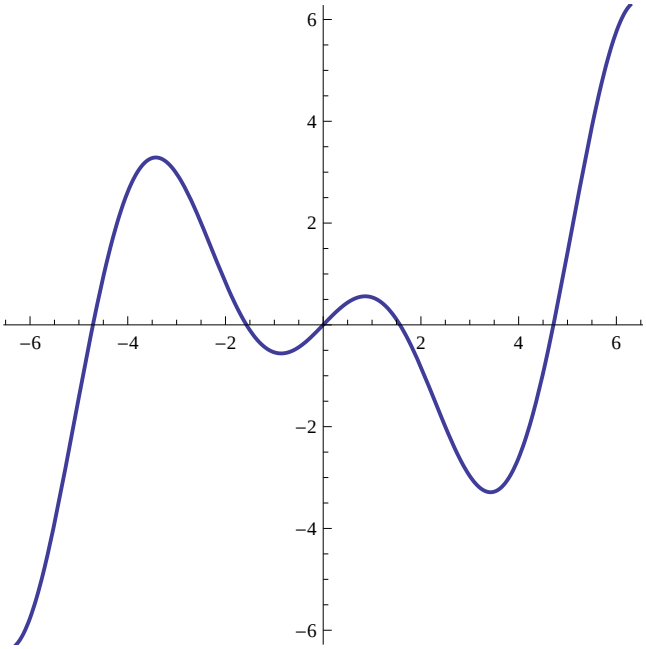
8. Find the equilibrium solutions of the differential equation $x' = x^2 - 4x - 5$. You can do this algebraically and visually by means of the direction field on appropriately chosen axes. Using the direction field examine the locations of the equilibrium solutions and explain differences between the equilibrium solutions. For example, which solution is *stable* or a *sink*, that is, it has the property that if an initial value is taken near the value of the equilibrium solution, the solution becomes asymptotically close to the equilibrium solution as t gets larger? Which equilibrium solution is *unstable* or a *source*, that is, it has the property that if an initial value is taken near the equilibrium solution, the solution with that initial value tends away from the equilibrium solution? Which is a *node*, that is, neither a source nor a sink? (See Project C in Chapter 1 in the text by Nagle, Saff, and Snider.) For some initial values close to the equilibrium values plot some solutions and see how the solutions behave. The following commands may help.

```
f[t_, x_] := x^2 - 4 x - 5;
dir = VectorPlot[{1, f[t, x]}, {t, fill, fill}, {x, fill, fill},
  VectorPoints → 15, Axes → True, Frame → False,
  VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.02, 1, None},
  PlotRange → {{fill, fill}, {fill, fill}}, AxesOrigin → {0, 0}, AxesLabel → {"t", "y(t)"}]
```

9. Check if the solution $x = 0$ of the non-autonomous differential equation $x' = -\frac{x}{1+t}$ has the property that if another solution is found whose initial value at t=0 is close to 0 then that solution is asymptotically close to $x = 0$ if t is large. Can any of your solutions ever intersect for large t?

10. Examine the plot of the function x cos x between -2π and 2π. What are all the equilibrium solutions of x' = x cos x between -2π and 2π? By examining the graph of the function, characterize each of the equilibrium solutions as to being sources, sinks, or nodes. Verify your finding by looking at the direction field and some representative solutions.

```
Plot[x Cos[x], {x, -2 π, 2 π}, PlotRange → {-2 π, 2 π}, AspectRatio → 1, PlotStyle → Thick]
```



3.6 Dependence on Parameters

We can observe changes in parameter values to understand how changes in a parameter affect solutions to a given differential equation. For example, how would a change in the initial value change the appearance of the solution? In the repetitive loop below, we consider the differential equation $\frac{dx(t)}{dt} = -x(t)(x(t) - 2)$ and vary the initial condition $x(0) = 2 - \frac{1}{\ln n}$ using the parameter n, and temporarily record a plot of each solution as a frame.

```
plotlist = {};
For[n = 2, n ≤ 25, n++,
  soln = Flatten[DSolve[{x'[t] == -x[t] * (x[t] - 2), x[0] == 2 - 1 / Log[n]}, x[t], t]];
  AppendTo[plotlist, Plot[x[t] /. soln, {t, -1, 5},
    PlotRange → {0, 2}, PlotStyle → {Hue[Random[]], Thick}]]];
```

We animate the frames by using the **ListAnimate** command.

```
ListAnimate[plotlist]
```

3.7 Exercises

11. Consider an equation given by $\frac{dx(t)}{dt} = \frac{t}{20} + \sin(n\,t)$ where we have introduced a parameter n. Let $x(0) = 1$ and plot animated frames of solutions as n varies, say from 1 to 100. (This might take a minute if your computer is slow.) What is the effect of changing n? Can you explain the behavior of the solution from the equation?

12. A model for the alligator population y at time t on the grounds of the Kennedy Space Center is given by the logistic equation $\frac{dy(t)}{dt} = -\frac{y(y-1500)}{3200}$ where *t* is measured in years. If hunters are allowed to kill *s* alligators per year, the equation would need to be modified as follows: $\frac{dy(t)}{dt} = -\frac{y(y-1500)}{3200} - s$. (See Nagle, Saff, and Snider Text, Project D, Chapter 1) Using code similar to that above we can show how the solutions change as the initial values change. Below we have taken *s* to be 100.

```
plotlist = {};
s = 100;
For[n = 1, n ≤ 20, n++,
  soln =
   Flatten[DSolve[{y'[t] == -y[t] * (y[t] - 1500) / 3200 - s, y[0] == 100 + n * 75}, y[t], t]];
  AppendTo[plotlist, Plot[y[t] /. soln, {t, 0, 5}, PlotRange -> {-200, 1600},
    PlotStyle -> {Hue[Random[]], Thick}]];
```

CAUTION: For certain values of the parameters m and s, *Mathematica* will generate solutions with complex values and empty plots. Consequently, when the **Plot** command is issued, the *Mathematica* program will produce a lot of error messages and no plot.

(a) Click on the graph and start the animation by using the **ListAnimate** command. Slow the animation down if necessary or look at one frame at a time. Determine from what you see, the equilibria and determine whether they are sinks or sources. By changing the function *f* a bit and maybe the range of the index *m*, you should be able to estimate the equilbria to within 10.

(b) Check your the values of your equilibria of (a) by finding the exact values algebraically.

(c) Experimentally, find an approximate value of *s* for which there is only one equilibrium value. (Hint: At this value, the graphs should only change from increasing to decreasing (or vice versa) only once.) What is the significance of this value of *s* on the alligator population?

Laboratory Four

Picard's Method

(Chapter 1, Project B of the Nagle/Saff/Snider text)

Picard's Method is a procedure whereby a sequence of functions is generated by an iterative process that converge to a solution of a given first-order differential equation. In the procedure the differential equation is converted to an integral equation that is used to generate the sequence of functions. This lab details the process.

The objectives of this lab are as follows.

1. To introduce Picard's method in a manner accessible to students.

2. To develop a *Mathematica* implementation of Picard's method.

3. To motivate discussion of the existence theory for Initial Value Problems.

Primary *Mathematica* commands used in this lab are **`Plot`**, **`VectorPlot`**, **`Show`**, and **`RSolve`**.

4.1 Background

First-order initial-value problems of the form $\frac{dy}{dx} = f(x, y),\ y(x_0) = y_0$ can be rewritten as an integral equation

$$y(x) = y_0 + \int_{x_0}^{x} f(t, y(t))\, dt$$

This integral formulation can be used to construct a sequence of approximate solutions to the IVP. The basic idea is that given an initial guess of the approximate solution to the IVP, say $\phi_0(x)$, an infinite sequence of functions, $\{\phi_n(x)\}$, is constructed according to the rule

$$\phi_{n+1}(x) = y_0 + \int_{x_0}^{x} f(t, \phi_n(t))\, dt$$

That is, the n^{th} approximation is inserted into the right-hand side of the integral equation in place of the exact solution *y(x)* and used to compute the $(n+1)^{st}$ element of the sequence. This process is called Picard's Method.

Without more knowledge about the solution to the initial-value problem, generally we take the initial solution to be $\phi_0(x) = x_0$.

One of the main applications of Picard's Method is in the proof of the existence and uniqueness results for first-order initial-value problems. This proof will not be considered in this lab but rather we will focus on gaining an understanding of Picard's Method through the consideration of a few exercises.

Let's consider an example. We will use Picard's Method with $\phi_0(x) = 1$ to obtain the next four successive approximations of the solution to $\frac{dy}{dx} = y(x),\ y(0) = 1$.

We start with the given initial approximation to the solution

```
ϕ₀[x_] := 1; y₀ = 1;
```

and compute the next approximation according to the integral formula

```
ϕ1[x_] := y0 + ∫0^x ϕ0[t] dt
ϕ1[x]
1 + x
```

The same process is used to compute all subsequent approximations. The second approximation to the solution can be obtained by repeating this process.

```
ϕ2[x_] := y0 + ∫0^x ϕ1[t] dt
ϕ2[x]
```

$$1 + x + \frac{x^2}{2}$$

For each of the last two approximate solutions, the two commands will be combined into a single *Mathematica* command.

```
ϕ3[x_] := y0 + ∫0^x ϕ2[t] dt
ϕ3[x]
ϕ4[x_] := y0 + ∫0^x ϕ3[t] dt
ϕ4[x]
```

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6}$$

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24}$$

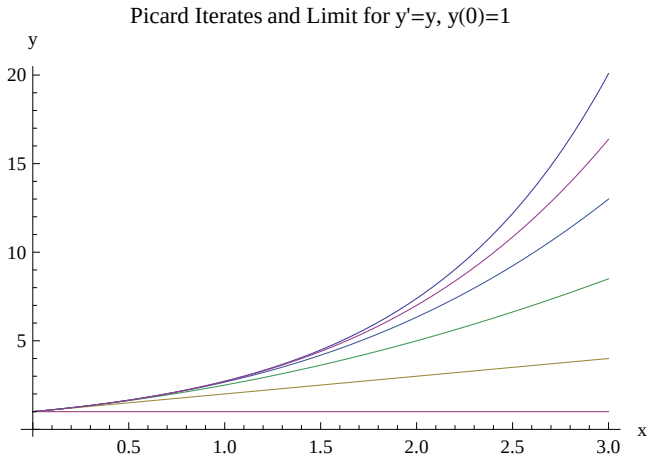
The pattern to the sequence of approximate solutions is obvious. Each iterate adds one new term. The new term for $\phi_n(x)$is $\frac{x^n}{n!}$. Thus, for each $n \geq 0$,

$$\phi_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}$$

These functions are the partial sums of the Maclaurin series for e^x.

From our knowledge of power series, we know that these partial sums converge to the exponential function for all real numbers x. This convergence can be demonstrated graphically.

```
Plot[{e^x, ϕ0[x], ϕ1[x], ϕ2[x], ϕ3[x], ϕ4[x]}, {x, 0, 3},
 AxesLabel → {"x", "y"}, PlotLabel → "Picard Iterates and Limit for y'=y, y(0)=1"]
```



To conclude this example, we note that the exponential function is a solution to the initial-value problem.

As another example we will use Picard's Method with $\phi_0(x) = 0$ to obtain the next three successive approximations of the solution to the nonlinear problem $\frac{dy}{dx} = 2x - y(x)^2,\ y(0) = 0$, on the interval [0,1].

These approximations could be obtained exactly as in the first example. However, it will ultimately be more efficient to create a *Mathematica* implementation of Picard's Method. The basic ingredients for Picard's Method are the function f on the right-hand side of the ODE, the initial condition, and the initial/previous approximate solution. Here is our version of the method.

```
Picard[f_, init_, phi_] :=
  Module[{x0 = init[[1]], y0 = init[[2]]}, y0 + Integrate[f[t, phi[x] /. x → t], {t, x0, x}]];
```

Note that the initial condition **init** is expected to be the ordered pair $\{x_0, y_0\}$, and **f** and **phi** are assumed to be *Mathematica* functions. The output from **Picard** is the next approximate solution as a *Mathematica* function so that the output is suitable for immediate use in another call to **Picard**.

To use **Picard** to provide the requested information in this example, let

```
Clear[x, f, phi, init];
f[x_, y_] := 2 x - y^2;
phi[x_] := 0;
init := {0, 0};
phi1[x_] := Picard[f, init, phi];
phi1[x]
```

x^2

This is simply a fancy way of saying that $\phi_1(x) = x^2$.

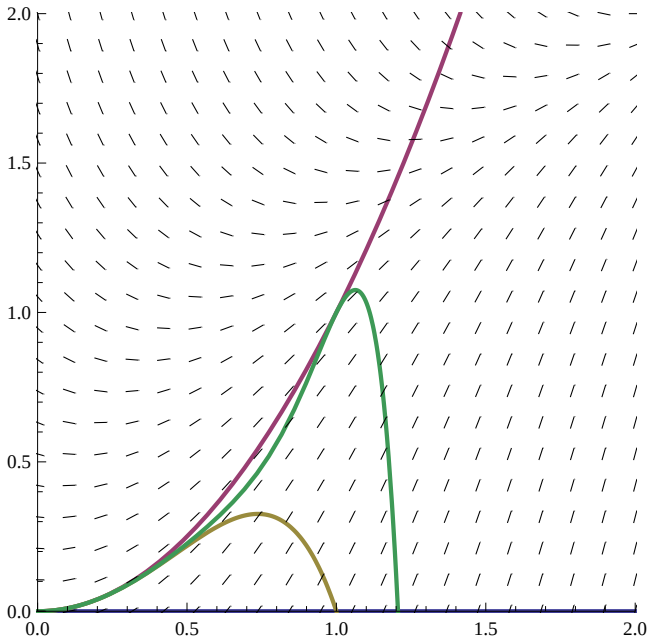
```
phi2[x_] := Picard[f, init, phi1];
phi2[x]
phi3[x_] := Picard[f, init, phi2];
phi3[x]
```

$$x^2 - \frac{x^5}{5}$$

$$x^2 - \frac{x^5}{5} + \frac{x^8}{20} - \frac{x^{11}}{275}$$

The convergence of these polynomials is not as obvious as the first example. Even from the graph it's difficult to determine the interval on which this integral exists. Including the direction field in the plot shows precisely how these approximate solutions must behave. Here is the graph that provides the most information. (The plots on [0,1] do not show too much difference between the successive approximations. Doubling the interval to [0,2] shows a little more of the interesting features of these solutions.)

```
gr = Plot[{phi[x], phi1[x], phi2[x], phi3[x]},
    {x, 0, 2}, PlotRange → {{0, 2}, {0, 2}}, PlotStyle → Thick];
dir = VectorPlot[{{1, 2 x - y^2}, {x, 0, 2}, {y, 0, 2},
    VectorPoints → 20, Axes → True, Frame → False,
    VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.015, 1, None},
    PlotRange → {{-12, 12}, {-12, 12}}, AxesOrigin → {0, 0}, AxesLabel → {"t", "y(t)"}];
Show[gr, dir, AspectRatio → 1]
```



4.2 Exercises

1. In the first example given above, $\frac{dy}{dx} = y(x),\ y(0) = 1$, take the initial guess to be the function $\phi_0(x) = 1 + x$. Use the procedure given above to find five iterates. Describe what happens. Is the sequence of iterates the same? Do they appear to converge to the same function? Repeat for another reasonable starting guess. You may want to increase the number of iterations to see convergence.

2. In the second example given above, $\frac{dy}{dx} = 2x - y(x)^2,\ y(0) = 0$, take the initial guess to be the function $\phi_0(x) = x$. Use the procedure above to find five iterates. Describe what happens. Is the sequence of iterates the same? Do they appear to converge to the same function?

3. Generate several Picard iterates for the initial value problem $\frac{dy}{dx} = 2x(y+1),\ y(0) = 0$,and determine to which function that they converge by examining the sequence of iterates. Check if you are correct by actually solving the equation analytically.

4. Compute some iterates of the function $\frac{dy}{dx} = e^x + y^2$ with the initial condition $y(0) = 0$. Determine from the iterates to what solution they converge.

4.3 A More Difficult Example

All of the examples above have involved intial-value problems that had unique solutions according to the Uniqueness and Existence Theorem for first-order equations. (See Nagle, Saff, and Snider, Section 1.2.) What happens when there is not a unique solution? Let's consider for example the IVP $\frac{dy}{dx} = 3y^{2/3},\ y(2) = 0$, which is not guaranteed to have a unique solution according to the theorem. Let's employ the **Picard** procedure to see what solution is generated. We will let the initial solution be the 0 function.

```
Clear[x, f, phi, init];
f[x_, y_] := 3 * y^(2 / 3);
phi[x_] := 0;
init := {2, 0};
phi1[x_] := Picard[f, init, phi];
phi2[x_] := Picard[f, init, phi1];
phi3[x_] := Picard[f, init, phi2];
phi4[x_] := Picard[f, init, phi3];
phi5[x_] := Picard[f, init, phi4];
phi5[x]
```

```
0
```

Clearly the solution generated is the 0 solution in this case. Let's however try another initial function $\phi_0(x) = x - 2$.

```
Clear[x, f, phi, init];
f[x_, y_] := 3 * y^(2 / 3);
phi[x_] := x - 2;
init = {2, 0};
phi1[x_] := Picard[f, init, phi];
phi2[x_] := Picard[f, init, phi1];
phi3[x_] := Picard[f, init, phi2];
phi4[x_] := Picard[f, init, phi3];
phi5[x_] := Picard[f, init, phi4];
PowerExpand[phi5[x]]
```

$$\frac{531\,441 \times 3^{32/81}\ (-2+x)^{665/243}}{665 \times 5^{52/81}\ 13^{4/9}\ 19^{8/27}\ 211^{2/3}}$$

Mathematica returns a function of the form $\phi_n(t) = c_n(t-2)^{r_n}$ for some constants c_n and r_n. That is, the successive iterations will be given by the formula

```
Clear[c, n, r];
```

```
phi[n_] = c[n] * (t - 2)^r[n]
```

```
(-2 + t)^r[n] c[n]
```

for some constants c_n and r_n. However, we know that $\phi_{n+1}(x) = y_0 + \int_{x_0}^{x} f(t, \phi_n(t))\,dt$, so that the derivative of $\phi_{n+1}$is $f(x, \phi_n(x))$. This will give us a formula for r_{n+1} in terms of $r_n$and c_{n+1}in terms of c_n.

```
D[phi[n + 1], t]
```

```
(-2 + t)^(-1+r[1+n]) c[1 + n] r[1 + n]
```

```
f[t, phi[n]]
```

```
3 ((-2 + t)^r[n] c[n])^(2/3)
```

These two expresions will be equal if the powers on $(t-2)$ and the coefficients are equal. Thus we have the recurrence relation

$$r_{n+1} = \frac{2}{3} r_n + 1$$

Since $r_0 = 1$, we can solve this recurrence relation to find:

```
RSolve[{r[n + 1] == (2 / 3) r[n] + 1, r[0] == 1}, r[n], n]
```

```
{{r[n] → 3^-n (-2^(1+n) + 3^(1+n))}}
```

Clearly the limit of the values of r_n is 3. To find the values of c_{n+1} we substitute a simple value in for t like 3 into $\phi_{n+1}'(t) = f(t, \phi_n(t))$getting

```
D[phi[n + 1], t] == f[t, phi[n]] /. t → 3
```

```
c[1 + n] r[1 + n] == 3 c[n]^(2/3)
```

If the $c_n$approach some limit c then since the r_n are approaching 3, we would get the relation $c = c^{2/3}$, so that if c is a real value it has the value of 1. We thus get that the limit function for the $\phi_n$is the function $y(x) = (x-2)^3$. Thus starting with the initial guess of $x - 2$ we get another solution for the IVP.

4.4 Exercises

5. Verify that the function $y(x) = (x-2)^3$ is a solution to the IVP.

6. (a) Use Picard's method (and the procedure **Picard** given above) with $\phi_0(0) = 1$ to compute the next six successive approximations of the solution to the nonlinear problem $\frac{dy}{dx} = 1 + y^2,\ y(0) = \frac{1}{2}$. Display the first four iterates through ϕ_4.

(b) To what do the constant, linear, and quadratic terms seem to converge from the appearance of the output? How many terms seem to be accurate after each iterate? How fast does the number of terms seem to be growing?

(c) Create plots of the each Picard iterate superimposed on the direction field.

(d) Produce and view an animation of the convergence of the Picard iterates to the solution of this IVP. What can you say about the existence of a solution to this IVP? Does it exist for all $x > 0$?

(e) Solve the IVP either by hand or with *Mathematica* and check if you wish to revise your answer to (d).

Laboratory Five

Applications of First-Order Differential Equations

(Chapter 3 of the Nagle/Saff/Snider text)

First-order differential equations can be used to model real world phenomena that vary with respect to time. This section looks at some examples of this modeling.

Our objectives in this section are as follows.

1. To use differential equations in modeling changes in population by means of the logistic population model.

2. To use differential equations in modeling various other growth and decay problems.

3. To use differential equations to model mixing problems.

Primary *Mathematica* commands used in this lab are **`Plot`**, **`VectorPlot`**, **`Show`**, and **`RSolve`**.

5.1 Population Growth

A simple way of modeling the growth (or decline) of a population N is to assume that the rate of change in the population at any time t is proportional to the quantity $N(t)$ that is present (or remaining). Formally, this rate of change is denoted as

$$\frac{dN}{dt} = rN$$

Separating the variables to solve for $N(t)$ with initial value $N(t_0) = N_0$ yields the *exponential* function $N(t) = N_0 e^{rt}$. The proportionality constant r is the difference in the per capita birth and death rates $(r - d)$. Growth occurs if $r > 0$and removal takes place when $r < 0$(what happens when $r = 0$?). Slowly reproducing species such as elephants, killer whales, and certain plants have a small r while rapidly reproducing organisms such as bacteria, lake trout, and small insects have a relatively large r. The above differential equation exhibits unbounded population growth at an exponential rate assuming unlimited resources.

Logistic growth, in comparison, limits the size of the population. Let constant k denote the maximum sustainable population size or the *carrying capacity* of the habitat of study. Then $k - N$ is the number of new individuals that the habitat can accept and $(k - N)/k$ is the percentage of k available for population growth. As the population N approaches the carrying capacity k, the percentage of k available for growth is small as a result of overcrowding and more competition for the limited resources results. Similarly, as $N \to 0$, $(k - N)/k \to 1$. Formally, we denote logistic population growth as

$$\frac{dN}{dt} = r\,\frac{k - N}{k}\,N$$

or equivalently as

$$\frac{dN}{dt} = r\left(1 - \frac{N}{k}\right)N = rN - cN^2$$

This equation succinctly shows that logistic population growth is the combination of the process of reproduction with the process of competition among pairs of individuals $N(t)^2$ by a factor of c.

We can modify the logistic equation by including the possibility that when the population is too small, say less than p, the inability to find a suitable mate leads to the extinction of the animal. The *modified logistic* equation is

$$\frac{dN}{dt} = \frac{rN(k-N)(N-p)}{kp} = rN\left(1-\frac{N}{k}\right)\left(\frac{N}{p}-1\right)$$

We can expand upon the exponential, logistic, and modified logistic by adding or subtracting a fixed amount $\alpha(t)$ for each time period. When $\alpha(t) > 0$, this can represent stocking, while if $\alpha(t) < 0$ this can represent harvesting or hunting.

5.2 Exercises

1. Generate the direction fields of the three models for population growth $\frac{dN}{dt} = rN$, $\frac{dN}{dt} = r\frac{k-N}{k}N$, and $\frac{dN}{dt} = \frac{rN(k-N)(N-p)}{kp}$ and record any observable differences. Take k to be 20, p to be 10 and r to be .8. What are the long term trends in the population in each case? Use the commands like those given below, filling in the appropriate data at the spots marked **fill**. Since **N** is reserved by *Mathematica*, we will use **y** instead.

```
dir = VectorPlot[fill, {t, fill, fill},
  {y, fill, fill}, PlotRange → {{fill, fill}, {fill, fill}},
  VectorPoints → fill, Axes → True, Frame → False, VectorStyle → {Black, Arrowheads[0]},
  VectorScale → {.015, 1, None}, PlotRange → {{fill, fill}, {fill, fill}},
  AxesOrigin → {0, 0}, AxesLabel → {"t", "N(t)"}]
```

2. Two bacteria are placed in a petri dish with unlimited resources and grow unchecked by the law of exponential growth. If they divide once every 20 minutes, how many bacteria are present after 1 day? Write an equation that solves this problem and use *Mathematica* to find the numerical answer.

3. Suppose that the population of a species of fish in a certain lake is hypothesized to grow according to a logistic model of population growth with growth rate $r = 0.3$and carrying capacity $k = 3000$. Assume initially there are 2500 fish of the species in the lake. Determine the correct differential equation for each of the scenarios below and in each case determine from the direction field of the differential equation the long term behavior of the fish population.

(a) Each year 150 fish are harvested from the lake. What is the general long term behavior?

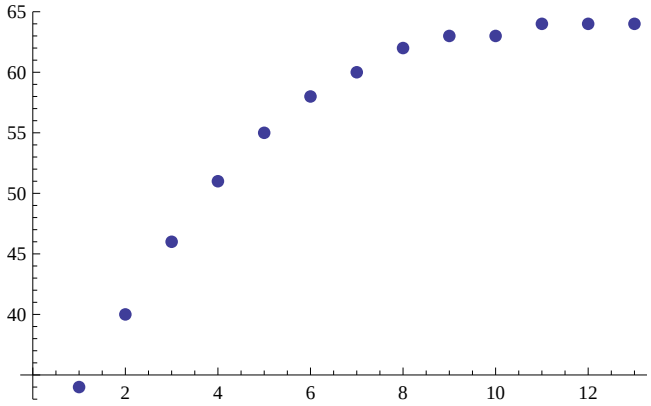
(b) Each year 25% of the fish are harvested from the lake. What is the general long term behavior?

(c) What is a maximum (within 50) safe fixed amount to harvest each year in order to assure that there will always be some fish in the lake?

4. The population of a species of birds in a natural preserve has been recorded each year for the past thirteen years. The populations are recorded in the following ordered pairs starting initially at year 1 as follows: (1,34), (2,40), (3,46), (4,51), (5,55), (6,58) (7,60), (8, 62), (9, 63), (10, 63), (11, 64), (12, 64), and (13, 64). We wish to determine what differential equation model most nearly models the behavior of this population.

```
popvector = {{1, 34}, {2, 40}, {3, 46}, {4, 51}, {5, 55},
   {6, 58}, {7, 60}, {8, 62}, {9, 63}, {10, 63}, {11, 64}, {12, 64}, {13, 64}}
ListPlot[popvector, PlotStyle → PointSize[0.02]]
```

```
{{1, 34}, {2, 40}, {3, 46}, {4, 51}, {5, 55}, {6, 58},
 {7, 60}, {8, 62}, {9, 63}, {10, 63}, {11, 64}, {12, 64}, {13, 64}}
```



(a) Which of the three population models -- exponential, logistic, modified logistic -- would you use for this model?

(b) Make a guess for the values of the parameter(s) in your model.

(c) Write the differential equation that you think models this population.

(d) Solve the differential equation using the **NDSolve** command and plot the solution together with the function *p*. Insert the correct values at the spots marked **fill**.

```
dir = VectorPlot[fill, {t, fill, fill},
    {y, fill, fill}, PlotRange → {{fill, fill}, {fill, fill}},
    VectorPoints → fill, Axes → True, Frame → False, VectorStyle → {Black, Arrowheads[0]},
    VectorScale → {.015, 1, None}, PlotRange → {{fill, fill}, {fill, fill}},
    AxesOrigin → {0, 0}, AxesLabel → {"t", "N(t)"}];
soln = NDSolve[{y'[t] == fill, y[0] == fill}, y, {t, fill, fill}];
g1 = Plot[y[t] /. soln, {t, fill, fill}, PlotStyle → Thick];
Show[g1, dir, AspectRatio → 1]
```

(e) If the plots do not reasonably coincide, adjust your model to get better coincidence.

(f) Compute what happens in the 14th year.

5.3 Other Models -- Exercises

Differential equations similar to population models can be used to model other phenomena such as heating, decay, annuities, mortgages, and spread of diseases. Here are three examples.

5. **Newton's Law of Cooling** states that the rate at which the temperature *T* of an object changes, $\frac{dT}{dt}$, is proportional to the difference between the temperature *N* of the surrounding medium and the temperature of the object . Mathematically we write this as $\frac{d}{dt}T(t) = k(N - t)$. Milk is brought out of the refrigerator compartment kept at 40 degrees into a warm room of unknown temperature. After 4 minutes the temperature of the milk is 54 degrees and after 6 minutes the temperature of the milk is 59 degrees.

(a) What is the temperature of the room? (For this you may wish to use the **Limit** command of *Mathematica*.)

(b) When will the milk be within 1 degree of the room temperature?

(c) Will it ever reach the room temperature?

Here are some *Mathematica* commands that may be useful in solving this problem. Insert the correct values at the spots marked **fill**.

```
Clear[k, N, t];
eqn = D[T[t], t] == k * (N - T[t]);
soln1 = DSolve[{eqn, T[0] == fill}, T[t], t]
Nvalue = Solve[(soln1[[1, 1, 2]] /. t → 4) == fill, N]
soln2 = Simplify[soln1 /. N → Nvalue[[1, 1, 2]]]
kvalue = Solve[(soln2[[1, 1, 2]] /. t → 6) == fill, k]
soln3 = Simplify[soln2 /. k → kvalue[[fill, fill, fill]]] // N
L = Limit[soln3[[1, 1, 2]], t → ∞]
Solve[soln3[[1, 1, 2]] == L - 1, t]
```

6. A high school mathematics teacher places $2000 into an **annuity** fund and then contributes $1800 per year into the fund for the next 30 years by making small weekly contributions. (We will assume weekly contributions are close enough to continuous deposits so that we may use a differential equation model.) The fund grows at a rate of 7.5% per year.

(a) Write a differential equation that models the growth of this fund.

(b) How much money will be in the fund after 30 years according to this model?

```
eqn = D[M[t], t] == fill;
soln = DSolve[{eqn, M[0] == 2000}, M[t], t]
tot1 = soln /. t → fill
```

(c) An alternate method of solving this problem is to use a difference equation since in actuality the additions are made to the annuity in discrete time intervals. (Difference equations are discussed in the text by Nagle, Saff, and Snider, but this discussion can be understood without knowing that section.) We can proceed as follows. How much money is in the fund after 30 years according to this model?

```
Clear[n];
addition = 1800 / 52; (* weekly addition to the annuity *)
rate = .075 / 52; (* weekly interest rate *)
reqn = a[n + 1] == a[n] + rate * a[n] + addition ; (* the recurrence relation *)
solnr = RSolve[{reqn, a[0] == 2000}, a[n], n]

tot2 = solnr /. n → fill
```

(d) Is your answer to part (c) equal to your answer to part (b) within a reasonable amount? If not, you should recheck your work. Which answer would you think is the most accurate?

(e) If at retirement after 30 years the money still grows at the same rate, but no new contributions are made into the fund, how much money *w* can be withdrawn each month before the money is exhausted in 20 years? (First write a differential equation that models what happens after retirement, again assuming that monthly withdrawals are close enough to continous withdrawals.)

```
eqn = D[MM[t], t] == fill;
soln = DSolve[{eqn, MM[0] == tot1}, MM[t], t]
tot3 = soln /. t → fill;
Solve[tot3 == 0, w]
```

(f) Now solve the same problem as in (e) with a difference equation, and see if you get essentially the same (within reason) result.

```
Clear[n];
rate = .075 / 52; (* weekly interest rate *)
reqn = a[n + 1] == fill ; (* the recurrence relation *)
solnr = RSolve[{reqn, a[0] == tot2}, a[n], n]
tot4 = solnr /. n → fill
Solve[tot4 == 0, w]
```

7. **Spread of Disease.** Suppose that an infectious disease is spreading among a population of *N* individuals. For simplicity we will assume that the incubation period is zero and that the changes in the progress of the disease is continuous. The population of *N* individuals is comprised of three types of individuals: the infected consisting of *i* individuals, the susceptible consisting of *s* individuals, and the removed individuals (immune, died, or removed into isolation) consisting of *r* individuals. The product of *r* times *s* represents the amount of contact between these two populations. Accordingly for suitable positive constants *a* and *b*, the following equations represent the population *N*.

$$N = i + s + r,\ \frac{ds}{dt} = -a\,s\,i,\ \frac{di}{dt} = -b\,i + a\,s\,i,\ \text{and}\ \frac{dr}{dt} = b\,i.$$

Dividing the third equation by the second equation we get the equation

$$\frac{di}{ds} = \frac{-b\,i + a\,s\,i}{a\,s\,i},$$

giving the rate of change of those infected with respect to those susceptible.

(a) Solve this equation for *i* in terms of *s*.

```
Clear[a, b, i, p, q, s];
eqn = D[i[s], s] == (b * i[s] - a * s * i[s]) / (a * s * i[s]);
soln = DSolve[{eqn, i[p] == q}, i[s], s]
```

$$\left\{\left\{\text{i[s]} \to \frac{\text{a p} + \text{a q} - \text{a s} - \text{b Log[p]} + \text{b Log[s]}}{\text{a}}\right\}\right\}$$

(b) The ratio b/a is important in this equation. So are the initial conditions *p* for *s* and q for *i*. What does the ratio represent? Be creative in some choices for these parameters and give the graphs of the solutions *i* with respect to *s*. What do the various graphs tell us about the relationship of those susceptible and those infected.

```
eqn1 = eqn /. a → fill /. b → fill;
soln1 = DSolve[{eqn1, i[fill] == fill}, i[s], s]
Plot[i[s] /. soln1, {s, 0.001, 2}]
```

5.4 Mixing Problems (Section 3.2) -- Exercises

8. Suppose a brine solution containing 2 kg of salt per liter runs into a tank initially filled with 500 L of water containing 50 kg of salt. The brine runs into the tank at the rate of 5 L/min. The mixture, kept uniform by stirring, is flowing out at rate of 5 L/min.

(a) Find the concentration, in kilograms per liter, of salt in the tank after 10 minutes.

```
eqn1 = D[A[t], t] == fill;
IC1 = A[0] == fill;
soln1 = DSolve[{eqn, IC1}, A[t], t];
p1 = Plot[A[t] /. soln1, {t, 0, 10}]
salt1 = soln1 /. t → fill;
conc1 = salt1/500
```

(b) After 10 minutes, a leak develops in the tank and an additional liter per minute of mixture flows out of the tank. What will be the concentration, in kilograms per liter, of salt in the tank 20 minutes after the leak develops.

For this problem the initial amount of salt is the amount calculated in the preceding problem after 10 minutes. Moreover the volume in the tank is decreasing by 1 liter per minute.

```
eqn2 = D[A[t], t] == 10 - 6 A[t]/(510 - t); IC2 = A[10] == salt1;
soln2 = DSolve[{eqn2, IC2}, A[t], t];
p2 = Plot[A[t] /. soln2, {t, 10, 530}]
salt2 = soln1 /. t → fill;
conc2 = salt2/(510 - fill)
Show[p1, p2]
```

(c) Is the graph totally correct? What happens when t is 510?

(d) If no leak had sprung, what would eventually be the amount of salt in the tank? Given that a leak did occur, what is the maximum amount of the salt in the tank? For the latter question, we can use the following commands.

```
dA = D[soln2[[1, 1, 2]], t];
tcrit = Solve[dA == 0, t];
maxsalt = soln2 /. t → tcrit
```

(e) Write a formula for the concentration in the tank for t > 20 and graph this solution. From the graph, what is the long term behavior of the concentration?

Laboratory Six

Further Applications of First-Order Differential Equations

(Section 3.3, Project D of Chapter 2, Project D of Chapter 3 of the Nagle/Saff/Snider text)

In this lab we will examine further applications of first-order differential equations. These supplement the applications given in the previous section. Applications included are the snowplow problem, aircraft guidance, and uses of Newton's law of cooling in the heating and cooling of buildings.

Let's begin with the snowplow problem.

6.1 Snowplow Problem (Project D of Chapter 2)

One morning it began to snow very hard and continued snowing steadily throughout the day. A snowplow set out at 9:00 A.M. to clear a road, clearing 2 miles by 11:00 A.M. and an additional mile by 1:00 P.M. At what time did it start snowing?

To construct a model for this problem, we must consider the rate of snow accumulation and the rate of the snowplow. We will assume that it is snowing at a constant rate (since is is snowing steadily) and also we will assume that the rate at which a snow plow can clear a length of road is inversely proportional to the weight of the snow being cleared. We will assume that the snow has a constant density and hence the weight of the snow is proportional to the volume of the snow, which we will assume is proportional to the height of the snow.

Let $s(t)$ represent the depth of the snow and $p(t)$ represent the distance plowed as functions of time t. Let $t = 0$ represent 9 A.M. and let $t = -T$ be the time it started to snow. So $t = 2$ represents 11 A.M. and $t = 4$ represents 1 P.M. The problem specifies the following: $s(-T) = 0$, $p(0) = 0$, $p(2) = 2$, and $p(4) = 3$. We can then solve the problem with the following set of differential equations.

```
Clear[a, b, p, s, t];
eqn1 = D[s[t], t] == a;
eqn2 = D[p[t], t] == b / s[t];
```

We now solve the differential equations for the functions s and p in terms of t.

```
soln1 = DSolve[{eqn1, s[-T] == 0}, s[t], t]
```

$$\{\{s[t] \to a\,t + a\,T\}\}$$

```
neweqn2 = eqn2 /. soln1
```

$$\left\{p'[t] == \frac{b}{a\,t + a\,T}\right\}$$

```
soln2 = DSolve[{neweqn2, p[0] == 0}, p[t], t]
```

$$\left\{\left\{p[t] \to \frac{-b\,\text{Log}[T] + b\,\text{Log}[t + T]}{a}\right\}\right\}$$

```
t1 = 0 == soln2[[1, 1, 2]] /. t → 0;
t2 = 2 == soln2[[1, 1, 2]] /. t → 2;
t3 = 3 == soln2[[1, 1, 2]] /. t → 4;
set1 = Solve[{t1, t2, t3}, {a, b, T}]
set2 = set1 // N
```

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

Solve::svars: Equations may not give solutions for all "solve" variables. ≫

$$\left\{\left\{T \to -1 + \sqrt{5},\ a \to b \operatorname{Log}\left[\frac{1}{2}\left(1 + \sqrt{5}\right)\right]\right\}\right\}$$

```
{{T → 1.23607, a → 0.481212 b}}
```

```
set2[[1, 1, 2]]
```

```
1.23607
```

6.2 Exercises

1. This means that it started to snow at 9 - T A.M. What time is this?

2. What is the actual time when it began to snow? Fill in the appropriate data in the spots labelled **fill**.

```
hour = 9 - fill
minutes = fill
seconds = fill
```

6.3 Aircraft Guidance in a Crosswind (Project D of Chapter 3)

An aircraft flying under the guidance of a nondirectional beacon (a fixed radio transmitter, abbreviated NDB) moves so that its longitudinal axis always points toward the beacon. A pilot sets out toward an NDB from a point at which the wind is at right angles to the initial direction of the aircraft; the wind maintains this direction. Assume the windspeed and the speed of the aircraft through the air (its "airspeed") remains constant.

By placing the start of the flight at (2,0) and the destination at (0,0), set up a differential equation describing the flight of the aircraft's path over the ground and solve the equation for the path.

For any position on the path of the aircraft, let ***T*** be the tangent vector, ***W =*** $(0, W)$ be the constant vector representing the wind, and ***A*** the vector pointing from the aircraft toward the beacon. Then ***T = A + W***. In terms of vector components, we then have the equation $\left(\frac{dx}{dt}, \frac{dy}{dt}\right) = (A_1, A_2 + W)$, where $\boldsymbol{A} = (A_1, A_2)$. Let k be the constant length of the vector ***A***. From these relationships, we get

```
Clear[W, k];
eqn = D[y[x], x] == y[x] / x - (W * Sqrt[x^2 + y[x]^2]) / (k * x)
soln = DSolve[eqn, y[x], x]
IC = y[2] == 0;
soln = DSolve[{eqn, IC}, y[x], x]
```

$$y'[x] == \frac{y[x]}{x} - \frac{W\sqrt{x^2 + y[x]^2}}{k\,x}$$

$$\left\{\left\{y[x] \to x\,\text{Sinh}\left[C[1] - \frac{W\,\text{Log}[x]}{k}\right]\right\}\right\}$$

Solve::ifun:

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

$$\left\{\left\{y[x] \to x\,\text{Sinh}\left[\frac{W\,\text{Log}[2]}{k} - \frac{W\,\text{Log}[x]}{k}\right]\right\}\right\}$$

6.4 Exercises

3. Show how we obtained the differential equation that we used.

4. Insert several values for *W* and *k* and determine the resulting solutions. Plot the solutions and see if they are what you expected.

6.5 A Different Equation for Aircraft Guidance in a Crosswind (Project D of Chapter 3)

Now we will let γ_1 represent the ratio $\frac{W}{k}$ of the wind speed to the airspeed in the differential equation to get a slightly different form of the equation. Substitute different values for this ratio and determine the resulting solution.

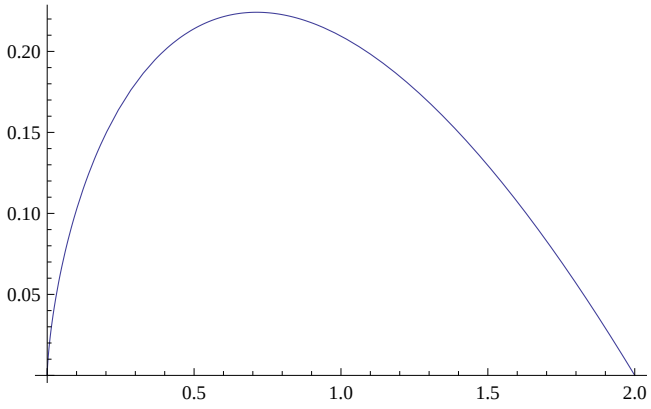
```
eqn = ∂x y[x] == y[x]/x - γ1 Sqrt[x^2 + y[x]^2]/x /. γ1 → 0.3
IC = y[2] == 0;
soln = DSolve[{eqn, IC}, y[x], x]
Plot[y[x] /. soln, {x, 0, 2}]
```

$$y'[x] == \frac{y[x]}{x} - \frac{0.3\sqrt{x^2 + y[x]^2}}{x}$$

Solve::ifun:

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

```
{{y[x] → x Sinh[0.1 (2.07944 - 3. Log[x])]}}
```



6.6 Exercises

5. Plot solution curves for other values of the parameter γ_1 getting closer to 1. Note the values on the *y*-axis in each case. What do you notice about these values as γ_1 gets closer to 0?

6. What happens (physically) when $\gamma_1 = 1$or $\gamma_1 > 1$?

6.7 Heating and Cooling of Buildings (Section 3.3)

A model for the change in temperature in the building is given by the differential equation

$$\frac{dT}{dt} = K[M(t) - T(t)] + H(t) + U(t)$$

where *T* represents the temperature of the building, $M(t)$ represents the outside temperature, $H(t)$ is the rate of change of temperature produced by people, lights, and machines inside the building, and $U(t)$is the rate of increase or decrease of temperature produced by a furnace or heat pump or an air conditioner. The parameter *K* is the proportionality constant given by Newton's Law of Cooling. We may take $M(t)$ to be a sinusoidal function that models the change of outside temperature over a 24 hour period, that is $M(t) = M_0 - A\cos(\pi t/12)$, where M_0 is the average temperature and *A* is a positive constant. Note that the period of M(t) is 24 hours.

Let's first take the case where the function $U(t) = K_U[T_D - T(t)]$ where T_D is the desired temperature in the building and K_U is a proportionality constant. Then we get the following equation.

```
Clear[A, H, K, T]; eqn = D[T[t], t] == K * (M[0] - A * Cos[π * t / 12] - T[t]) + H[t] + K_U * (T_D - T[t])
```

$$T'[t] == H[t] + K\left(-A\,\text{Cos}\left[\frac{\pi\, t}{12}\right] + M[0] - T[t]\right) + K_U\,(T_D - T[t])$$

If we assume the outside average temperature is 7°C, the desired temperature in the building is 24°C, $K + K_U = 2$, $1/4 < K < 1/2$, and the variation on temperature is 14°C each each day (so $A = 7$), let's see how long it takes for the inside temperature to get close to the desired temperature for various values of K. (The value $1/K$ is called the time constant for the building and the value $1/(K + K_U)$ is called the time constant for the building with heating and cooling.) We also wish to know, which buildings are better insulated, those with small K or large K. We will assume $H(t) = 0$, that is, there are no people or machines running in the building. Let's also assume the initial temperature of the building is 20°C.

```
eqn1 = eqn /. {H[t] → 0, K_U → 2 - K, M[0] → 7, T_D → 24, A → 7}
```

$$T'[t] == (2 - K)\,(24 - T[t]) + K\left(7 - 7\,\text{Cos}\left[\frac{\pi\, t}{12}\right] - T[t]\right)$$

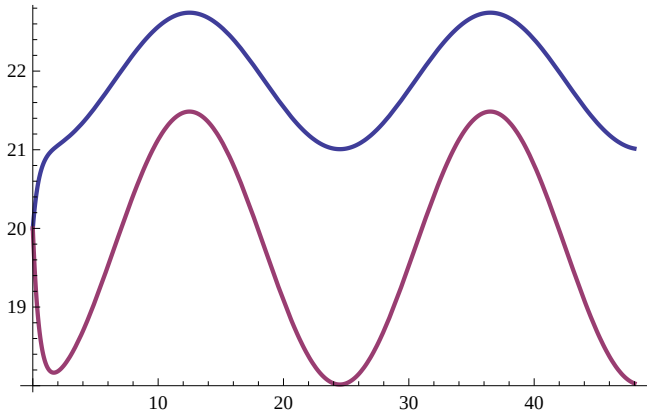
```
IC = T[0] == 20;
K = 1 / 4;
soln1 = Simplify[DSolve[{eqn1, IC}, T[t], t]]
```

$$\left\{\left\{T[t] \to -\frac{1}{8\left(576 + \pi^2\right)}\left(-100\,800 + 4608\,e^{-2t} - 175\,\pi^2 + 15\,e^{-2t}\,\pi^2 + 4032\,\text{Cos}\left[\frac{\pi\, t}{12}\right] + 168\,\pi\,\text{Sin}\left[\frac{\pi\, t}{12}\right]\right)\right\}\right\}$$

```
K = 1 / 2;
soln2 = Simplify[DSolve[{eqn1, IC}, T[t], t]]
```

$$\left\{\left\{T[t] \to \frac{1}{2304 + 4\,\pi^2}\left(45\,504 + 4608\,e^{-2t} + 79\,\pi^2 + e^{-2t}\,\pi^2 - 4032\,\text{Cos}\left[\frac{\pi\, t}{12}\right] - 168\,\pi\,\text{Sin}\left[\frac{\pi\, t}{12}\right]\right)\right\}\right\}$$

```
Plot[{T[t] /. soln1, T[t] /. soln2}, {t, 0, 48}, PlotStyle → Thick]
```



6.8 Exercises

7. Which value of K gives a more comfortable temperature sooner? From the graphs of the two functions, which value of K represents the better insulation? Why?

8. See what happens if there is a greater or less variation of temperature outside. Write your conclusions.

6.9 Further Exploration -- Heating and Cooling of Buildings (See Section 3.3, p. 101 of Nagle/Saff/Snider text)

We will now take the case where the function $U(t)$ represents a more realistic constant rate of heat flow, say $U(t) = 20$ units per hour if $T(t) < T_D$ and constant rate of cooling -5 units per hour if $T(t) > T_D + 1$. In between T_D and $T_D + 1$ there is no artificial heating or cooling by a furnace, heat pump, or air conditioner. To solve this new setup, we will start with an initial temperature. We will then heat the building to a small ϵ above the desired temperature T_D, or cool it to -ϵ below the desired temperature, or neither heat nor cool if the temperature is between T_D and $T_D + 1$. Again we will assume $H(t) = 0$, and that the average outside temperature is 7 which varies 14 degrees each day, although these hypotheses could be changed. Depending on whether the temperature is below T_D, between T_D and $T_D + 1$, or above $T_D + 1$, we will need to solve different differential equations (depending on the value of $U(t)$) and piece the solutions together to get a continuous curve for the temperature.

To implement this scenario, we will write a procedure called Temperature that solves various differential equations depending if $U(t) = 20$, $U(t) = -5$, or $U(t) = 0$. To use the procedure we will need to input the time constant of the building (a number between 2 and 4), the tolerance for the temperature of the building, the initial value for $t(0)$, the initial temperature of the building, the desired temperature in the building, and the number of iterations (the number of times for the execution of the loop, the more the longer the time span under consideration).

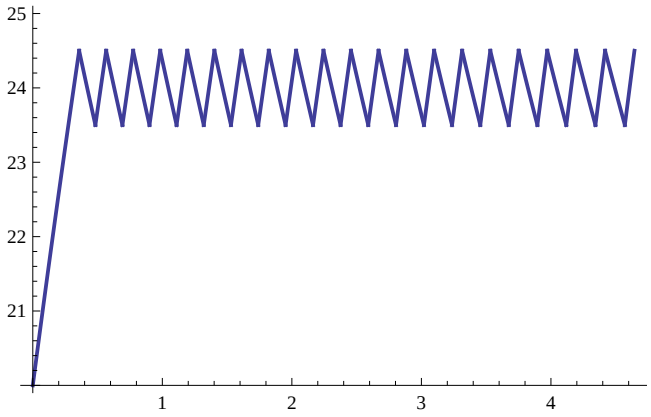
To execute this procedure we now need to type the command

```
Temperature[tcon, tol, t0, temp0, TD, iter]
```

where we insert values for the time constant of the building (a number between 2 and 4), the tolerance for the temperature of the building, the initial value for $t(0)$, the initial temperature of the building, the desired temperature in the building, and the number of iterations (the number of times for the execution of the loop, the more the longer the time span under consideration).

```
Temperature[tcon_, tol_, t0_, temp0_, tempd_, iter_] :=
  Module[
   {k = 1 / tcon, i, B = temp0, IC, eqn, timelist = Table[t[j], {j, 0, iter + 1}], plotlist = {}},
   t[0] = t0;
   For[i = 0, i ≤ iter,
    If[B < tempd,
     IC = T[t[i]] == B;
     eqn = D[T[x], x] == k * (7 - 7 * Cos[Pi * x / 12] - T[x]) + 20;
     soln = DSolve[{eqn, IC}, T[x], x];
     t[i + 1] = FindRoot[soln[[1, 1, 2]] - tempd - tol, {x, t[i]}][[1, 2]];
     AppendTo[plotlist,
      Plot[soln[[1, 1, 2]], {x, t[i], t[i + 1]}, PlotStyle → {Thickness[0.006]}]];];
    If[tempd ≤ B && B ≤ tempd + 1,
     IC = T[t[i]] == B;
     eqn = D[T[x], x] == k * (7 - 7 * Cos[Pi * x / 12] - T[x]);
     soln = DSolve[{eqn, IC}, T[x], x];
     t[i + 1] = FindRoot[soln[[1, 1, 2]] - tempd + tol, {x, t[i]}][[1, 2]];
     AppendTo[plotlist,
      Plot[soln[[1, 1, 2]], {x, t[i], t[i + 1]}, PlotStyle → {Thickness[0.006]}]];];
    If[tempd + 1 < B,
     IC = T[t[i]] == B;
     eqn = D[T[x], x] == k * (7 - 7 * Cos[Pi * x / 12] - T[x]) - 5;
     soln = DSolve[{eqn, IC}, T[x], x];
     t[i + 1] = FindRoot[soln[[1, 1, 2]] - tempd, {x, t[i]}][[1, 2]];
     AppendTo[plotlist,
      Plot[soln[[1, 1, 2]], {x, t[i], t[i + 1]}, PlotStyle → {Thickness[0.006]}]];
    ];
    B = soln[[1, 1, 2]] /. x → t[i + 1];
    i++];
   plotlist];

Temperature[3, 1/2, 0, 20, 24, 40];
Show[%, PlotRange → {20, 25}]
```



6.10 Exercises

9. Change the tolerance a little to see the effect on the graph.

10. Change the variation of the outside temperature a bit to see the effect.

11. Change the average outside temperature to see the effect on the graph.

12. For tolerances of $\epsilon = \frac{1}{2}$and $\epsilon = \frac{9}{10}$with the same outside temperature and variation of outside temperature over 24 hours, which tolerance uses the most energy? Can you determine numerical values for each amount of energy? (Hint: The values of $t(i)$represent the times when the cooling or heating units turn on or off.)

Laboratory Seven

Higher Order Differential Equations

(Chapters 4 and 6 of the Nagle/Saff/Snider text)

Mathematica can be used to solve second-order and higher-order differential equations - both homogenous and nonhomogeneous - by means of built in commands. For equations for which analytic solutions are not available by these commands, numerical approximations of solutions can be given. *Mathematica* can also be used to solve nonhomogeneous equations by means of either the method of variations of parameters or the method of undetermined coefficients.

The objectives of this section are as follows.

1. To learn how to use the *Mathematica* **DSolve** command to solve second (and higher) order differential equations and initial-value problems.

2. To learn how to use *Mathematica* to generate numerical approximations to solutions of higher-order differential equations.

3. To learn how to use *Mathematica* to solve nonhomogeneous equations by the variation of parameters method.

Key *Mathematica* commands to be used are **DSolve**, **NDSolve**, and **ComplexExpand**.

7.1 Using the DSolve Command

Example 1. Let's take a second-order differential equation with constant coefficients and let *Mathematica* solve it with the **DSolve** command. Consider the equation

$$\frac{d^2 y}{dt^2} + 6\frac{dy}{dt} - 2y = t^3 + \cos(t)$$

In *Mathematica* we would enter it as follows.

```
Clear[y];
soln = DSolve[y''[t] + 6 y'[t] - 2 y[t] == t^3 + Cos[t], y[t], t];
Simplify[soln]
```

$$\left\{\left\{y[t] \to \frac{1}{30}\left(-15\left(180 + 57\,t + 9\,t^2 + t^3 - 2\,e^{-\left(3+\sqrt{11}\right)t}\,C[1] - 2\,e^{\left(-3+\sqrt{11}\right)t}\,C[2]\right) - 2\,\text{Cos}[t] + 4\,\text{Sin}[t]\right)\right\}\right\}$$

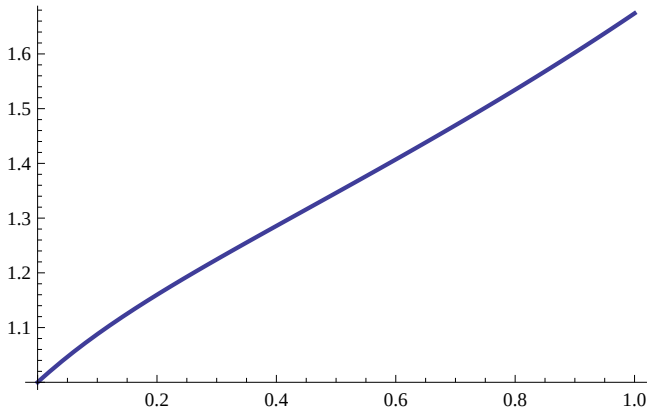
Let's add some initial values to get an initial-value problem.

```
soln = DSolve[{y''[t] + 6 y'[t] - 2 y[t] == t^3 + Cos[t], y[0] == 1, y'[0] == 1}, y[t], t];
Simplify[soln]
```

$$\left\{\left\{y[t] \to \frac{1}{660}\left(-59\,400 + 30\,052\,e^{\left(-3+\sqrt{11}\right)t} + 9077\sqrt{11}\,e^{\left(-3+\sqrt{11}\right)t} + 30\,052\,e^{-\left(3+\sqrt{11}\right)t} - 9077\sqrt{11}\,e^{-\left(3+\sqrt{11}\right)t} - 18\,810\,t - 2970\,t^2 - 330\,t^3 - 44\,\text{Cos}[t] + 88\,\text{Sin}[t]\right)\right\}\right\}$$

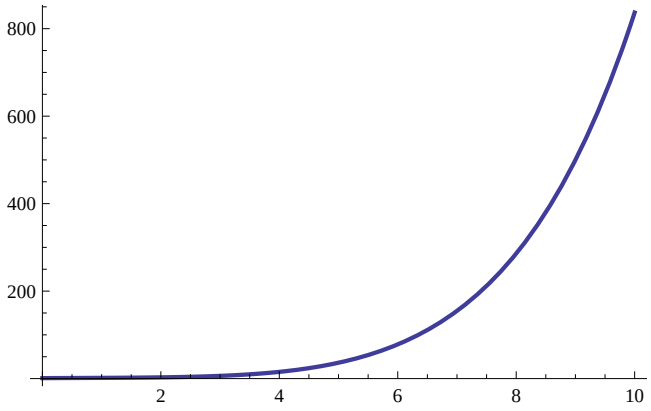
We can now graph the solution as well.

```
Plot[y[t] /. soln, {t, 0, 1}, PlotStyle → Thick]
```



A plot on a longer interval shows how the exponential term dominates the solution.

```
Plot[y[t] /. soln, {t, 0, 10}, PlotStyle → Thick]
```



Example 2. The same process can be used to solve higher-order initial-value problems as well. Let's see how this works on a third-order differential equation with initial conditions. Let's consider the third order equation

$$\frac{d^3 y}{dt^3} + \frac{d^2 y}{dt^2} + 3\frac{dy}{dt} - 5y = e^t$$

with initial conditions $y(0) = 0$, $y'(0) = 1$, and $y''(0) = -3$:

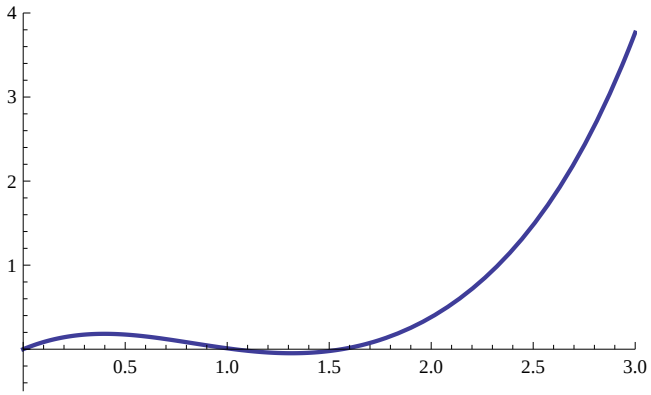
```
Clear[y];
soln = DSolve[y'''[t] + y''[t] + 3 y'[t] - 5 y[t] == Exp[t], y[t], t];
Simplify[soln]
```

$$\left\{\left\{y[t] \to \frac{1}{16} e^{-t}\left(e^{2t}(-1 + 2t + 16\,C[3]) + 16\,C[2]\,\text{Cos}[2t] + 16\,C[1]\,\text{Sin}[2t]\right)\right\}\right\}$$

```
Clear[y];
soln = DSolve[
   {y'''[t] + y''[t] + 3 y'[t] - 5 y[t] == Exp[t], y[0] == 0, y'[0] == 1, y''[0] == -3}, y[t], t];
Simplify[
 soln]
```

$$\left\{\left\{y[t] \to \frac{1}{16} e^{-t} \left(e^{2t} (-3 + 2t) + 3\cos[2t] + 10\sin[2t]\right)\right\}\right\}$$

```
Plot[y[t] /. soln, {t, 0, 3}, PlotRange → {{0, 3}, {-0.5`, 4}}, PlotStyle → Thick]
```



Example 3. How about if we have a differential equation that is not linear or does not have constant coefficients as do the previous examples? For example,

$$\frac{d^2}{dt^2} y(t) + y(t)^2 = \sin(t)$$

Let's see what happens.

```
Clear[y];
soln = DSolve[y''[t] + y[t]^2 == Sin[t], y[t], t];
Simplify[soln]
```

```
DSolve[Sin[t] == y[t]^2 + y''[t], y[t], t]
```

In this case *Mathematica* did not return a solution. We can find a numeric approximation to an initial value problem using this equation.

```
Clear[y];
soln = NDSolve[{y''[t] + y[t]^2 == Sin[t], y[0] == 0, y'[0] == 1}, y[t], {t, 0, 2}]
```

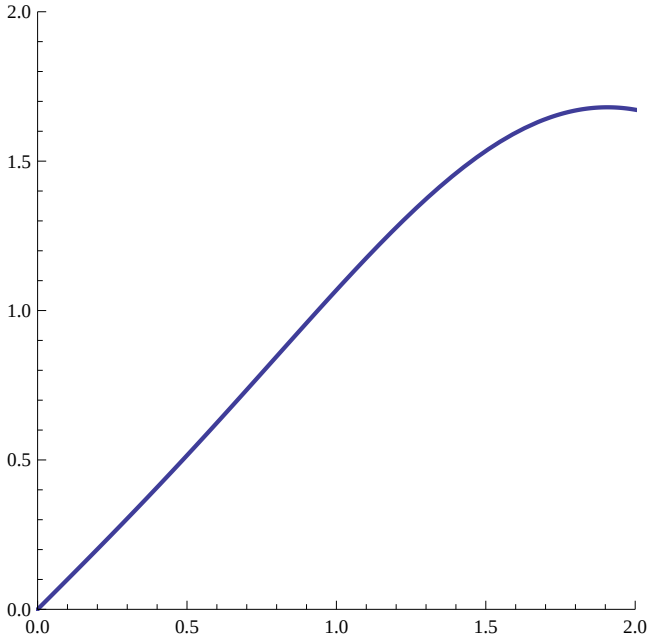
```
{{y[t] → InterpolatingFunction[{{0., 2.}}, <>][t]}}
```

This indicates that the numerical solution has been generated. Values of the solution can now by obtained and the graph of the solution (the numerical approximation of it) can be obtained with the **Plot** command as above.

```
y[t] /. soln /. t → 1
```

```
{1.0684}
```

```
Plot[y[t] /. soln, {t, 0, 2}, PlotRange → {{0, 2}, {0, 2}}, AspectRatio → 1, PlotStyle → Thick]
```



7.2 Exercises

1. In Example 1 above, from the solution generated by *Mathematica*, what is a fundamental set of solutions to the homogenous equation? Check your answer by finding the roots of the characteristic polynomial with the following code.

```
p := x^2 + 6 x - 2;
Solve[p == 0, x]
% // N
```

$$\left\{\left\{x \to -3 - \sqrt{11}\right\}, \left\{x \to -3 + \sqrt{11}\right\}\right\}$$

```
{{x → -6.31662}, {x → 0.316625}}
```

2. In Example 2 above, from the solution generated by *Mathematica*, what is a fundamental set of solutions to the homogenous equation? Check your answer by finding the roots of the characteristic polynomial with the code used in Exercise 1.

3. Solve the initial-value problem

$$\frac{d^3 y}{dt^3} + 2\frac{d^2 y}{dt^2} - \frac{dy}{dt} - 2y = e^t,\ y(0) = 0,\ y'(0) = 1,\ y''(0) = 1$$

using *Mathematica*.

4. **Transverse Vibrations of a Beam.** In the text by Nagle, Saff , and Snider, a fourth-order equation

$$\frac{d^4 y}{d t^4} - r^4 y = 0$$

is used to model transverse vibrations of a beam. The solution must satisfy the boundary conditions $y(0) = y(L) = 0$and $y'(0) = y'(L) = 0$ where L is the length of the beam. The problem is to find values of r for which this boundary-value problem has a nonzero solution. Use *Mathematica* to do it.

```
soln = DSolve[y''''[t] - r^4 * y[t] == 0, y[t], t]
```

```
{{y[t] → ℯ^(-r t) C[2] + ℯ^(r t) C[4] + C[1] Cos[r t] + C[3] Sin[r t]}}
```

```
y[t] /. soln
```

```
{ℯ^(-r t) C[2] + ℯ^(r t) C[4] + C[1] Cos[r t] + C[3] Sin[r t]}
```

```
eqn1 = First[y[t] /. soln /. t → 0] == 0
eqn2 = First[y[t] /. soln /. t → L] == 0
```

```
C[1] + C[2] + C[4] == 0
```

```
ℯ^(-L r) C[2] + ℯ^(L r) C[4] + C[1] Cos[L r] + C[3] Sin[L r] == 0
```

```
eqn3 = First[D[y[t] /. soln, t] /. t → 0] == 0
eqn4 = First[D[y[t] /. soln, t] /. t → L] == 0
```

```
-r C[2] + r C[3] + r C[4] == 0
```

```
-ℯ^(-L r) r C[2] + ℯ^(L r) r C[4] + r C[3] Cos[L r] - r C[1] Sin[L r] == 0
```

```
Solve[{eqn1, eqn2, eqn3, eqn4}, {C[1], C[2], C[3], C[4]}]
```

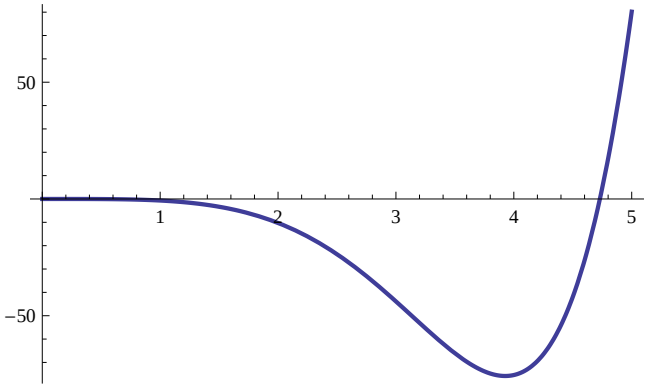
```
{{C[1] → 0, C[2] → 0, C[3] → 0, C[4] → 0}}
```

Although *Mathematica* generates only the trivial solution, let's look at little closer by means of a little linear algebra. We will generate the matrix of coefficients for the system of four equations treating $\cos(r L)$, $\sin(r L)$, $e^{r L}$, and $e^{-r L}$ as the coefficients and checking if and when there is a nontrivial solution.

```
A = {{1, 0, 1, 1}, {0, 1, 1, -1}, {Cos[r * L], Sin[r * L], Exp[r * L], Exp[-r * L]},
    {-Sin[r * L], Cos[r * L], Exp[r * L], -Exp[-r * L]}};
h = Simplify[Det[A]] /. r → a / L
```

```
-4 + 2 ℯ^(-a) (1 + ℯ^(2 a)) Cos[a]
```

```
Plot[h, {a, 0, 5}, PlotStyle → Thick]
```



What is the value of "a" for which the linear system of equations has a nonzero solution? (Fill in the correct data in the input below at the spots marked **fill**.)

```
FindRoot[fill == fill, {a, 5}]
```

From this result it is evident that for there to be a nontrivial solution where not all the c[n]'s are 0, we must have *r L* equal to this last answer. Now we can go back and compute the values of the c[n]'s in the general solution.

7.3 Further Considerations

Sometimes *Mathematica* solves the differential equation using the **DSolve** command, but the solution is not apparent. In this case we may need to solve the equation by the usual paper and pen techniques but let the computer do the calculation. Consider the following example.

Example 4. Solve the differential equation

$$\frac{d^4 y}{dt^4} + \frac{dy}{dt} + y = 0$$

```
DSolve[y''''[t] + y'[t] + y[t] == 0, y[t], t]
```

```
{{y[t] → ℯ^(t Root[1+#1+#1^4&,1]) C[1] + ℯ^(t Root[1+#1+#1^4&,2]) C[2] + ℯ^(t Root[1+#1+#1^4&,3]) C[3] + ℯ^(t Root[1+#1+#1^4&,4]) C[4]}}
```

This is the form of the solution, but as it stands it is unusable. Let's proceed then to use *Mathematica* to solve another way. We will solve the characteristic equation, get four roots, and substitute them in to the exponential function to find four linearly independent solutions.

```
p = r^4 + r + 1
```

```
1 + r + r^4
```

```
charroots = Solve[p == 0, r] // N
```

```
{{r → 0.727136 - 0.934099 i}, {r → 0.727136 + 0.934099 i},
 {r → -0.727136 - 0.430014 i}, {r → -0.727136 + 0.430014 i}}
```

```
soln1 = ComplexExpand[Exp[r * t] /. charroots[[1]]]
```

```
ℯ^(0.727136 t) Cos[0.934099 t] - i ℯ^(0.727136 t) Sin[0.934099 t]
```

```
soln2 = ComplexExpand[Exp[r * t] /. charroots[[2]]]
```

```
ℯ^(0.727136 t) Cos[0.934099 t] + i ℯ^(0.727136 t) Sin[0.934099 t]
```

```
soln3 = ComplexExpand[Exp[r * t] /. charroots[[3]]]
```

```
ℯ^(-0.727136 t) Cos[0.430014 t] - i ℯ^(-0.727136 t) Sin[0.430014 t]
```

```
soln4 = ComplexExpand[Exp[r * t] /. charroots[[4]]]
```

```
ℯ^(-0.727136 t) Cos[0.430014 t] + i ℯ^(-0.727136 t) Sin[0.430014 t]
```

To get four linearly independent solutions we take the real or complex parts of these solutions.

```
y1 = ComplexExpand[Re[soln1]]
```

```
ℯ^(0.727136 t) Cos[0.934099 t]
```

```
y2 = ComplexExpand[Im[soln2]]
```

$e^{0.727136\,t}\,\text{Sin}[0.934099\,t]$

```
y3 = ComplexExpand[Re[soln3]]
```

$e^{-0.727136\,t}\,\text{Cos}[0.430014\,t]$

```
y4 = ComplexExpand[Im[soln4]]
```

$e^{-0.727136\,t}\,\text{Sin}[0.430014\,t]$

The general solution is now a linear combination of these four functions. In the case of a nonhomogeneous equation, if we can determine the solution to the homogenous equation then we can use variation of parameters to obtain the solution of the nonhomogenous equation, although the attendant algebra becomes complicated for all but the lowest order equations.

7.4 Exercises

5. Solve the differential equation

$$\frac{d^2 y}{dt^2} - 2\sqrt{5}\,\frac{dy}{dt} + 14\,y = e^t$$

by means of variation of parameters. First find the roots of the characteristic equation and generate a fundamental solution set of the homogenous equation as above, then solve the system

$$y_1\,v_1' + y_2\,v_2' = 0$$
$$y_1'\,v_1' + y_2'\,v_2' = e^t$$

The following commands may be useful. Fill in the blanks marked fill with appropriate data.

```
p = fill;
charroots = Solve[p == 0, r]
soln1 = ComplexExpand[Exp[r * t] /. charroots[[1]]]
soln2 = ComplexExpand[Exp[r * t] /. charroots[[2]]]
y1 = ComplexExpand[Re[soln1]]
y2 = ComplexExpand[Im[soln2]]
eqn1 = fill == 0;
eqn2 = fill * dv1 + fill * dv2 == fill;
soln = Simplify[Solve[{eqn1, eqn2}, {dv1, dv2}]]
v1 = Integrate[soln[[1, 1, 2]], t]
v2 = Integrate[soln[[1, 2, 2]], t]
Simplify[v1 * y1 + v2 * y2] + c1 * y1 + c2 * y2
```

6. Repeat Exercise 5 but this time let *Mathematica* generate the fundamental solution set with the following commands.

```
eqn = D[y[t], {t, 2}] - 2 Sqrt[5] D[y[t], t] + 14 y[t] == 0
DSolve[eqn, y[t], t]
```

Laboratory Eight

An Application of Second Order Equations

(Sections 4.1-4.3 and 4.8 of the Nagle/Saff/Snider text)

In this section we will examine **harmonic** oscillation. We will model with differential equations the motion of a mass-spring system.

Our objectives are as follows.

1. To determine the effect of parameters on the solutions of differential equations.

2. To determine the behavior of the mass-spring from the graph of the solution.

3. To determine the effect of the parameters on the behavior of the mass-spring.

Primary *Mathematica* commands to be used are the **DSolve** and the **D** commands.

8.1 Mass-Spring without Damping

Motion can be modeled by specifying the displacement and velocity. The basic equation of the motion of the mass-spring system with mass m, damping coefficient c (frictional forces in the spring, air resistance), and spring constant k, assuming a linear-spring and linear friction, is

$$m\,x''(t) + c\,x'(t) + k\,x(t) = 0$$

If friction or other resistive forces are negligible, that is $c = 0$, then the equation above can be re-written as

$$x''(t) + \omega^2\,x(t) = 0$$

where $\omega^2 = k/m$. This models simple harmonic motion because the solution to this equation is

```
DSolve[x''[t] + ω^2 * x[t] == 0, x[t], t]
```

```
{{x[t] → C[1] Cos[t ω] + C[2] Sin[t ω]}}
```

which is a linear combination of the cosine and sine functions in terms of arbitrary constants `C[1]` and `C[2]`. To derive a specific solution to this equation, the initial displacement $x(0) = x_0$ and initial velocity $x'(0) = v_0$ must be provided.

Consider as an example, a mass of 1 kg attached to the end of a spring whose spring constant $k = 4$. The harmonic motion of the mass can be formulated as follows.

```
m = 1;
k = 4;
ω = Sqrt[k / m]
```

```
2
```

```
D[x[t], {t, 2}] + ω^2 * x[t] == 0
```

$$4\,x[t] + x''[t] == 0$$

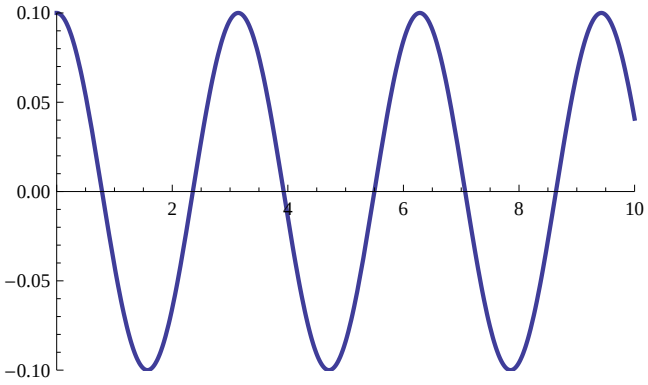
Assume that the mass initially is stretched 10 cm and then released. Then the initial position in terms of meters is $x(0) = 0.1$ meter and the initial velocity is zero, so $x'(0) = 0$. The specific solution with these initial conditions is

```
soln = DSolve[{x''[t] + ω^2 * x[t] == 0, x[0] == 1 / 10, x'[0] == 0}, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{1}{10}\,\text{Cos}[2\,t]\right\}\right\}$$

This solution plotted for a period of time is illustrated here.

```
Plot[x[t] /. soln, {t, 0, 10}, PlotRange → {{0, 10}, {-0.1, 0.1}}, PlotStyle → Thick]
```



8.2 Exercises

From the graph above answer the following questions.

1. What is the period of the motion?

2. When will this mass-spring system come to rest? If never, why?

3. What is the maximum displacement of the mass?

4. What is the maximum velocity attained by the mass, and when is this maximum velocity attained?

By experimenting with other values for the parameters *k* and *m*, answer the following.

5. How do physical parameters of the size of the mass and the stiffness of the spring affect the motion? Show the graphs which illustrate your assertions.

8.3 Mass-Spring with Damping

Let's now consider the case where there is some **damping** present. To this end, consider the equation

$$x''(t) + c\,x(t) + x(t) = 0$$

where we will take $x(0) = 1$ and $x'(0) = 0$. We assume that the time t is always nonnegative.

```
D[x[t], {t, 2}] + c * D[x[t], t] + x[t] == 0
```

```
x[t] + c x'[t] + x''[t] == 0
```

8.4 More Exercises

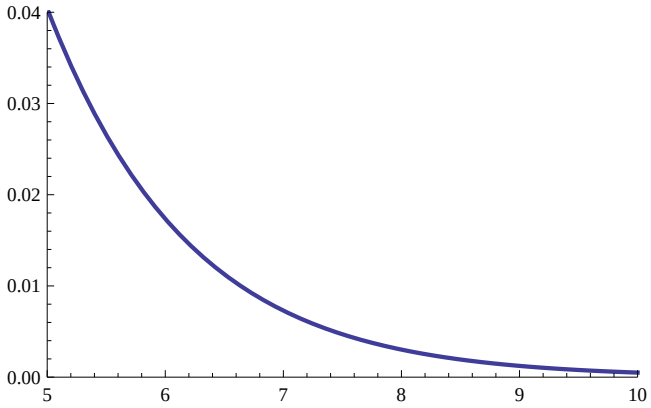
6. By graphing the solution in the case when c = 2, determine the value of time t, call it $t2$, after which the solution always satisfies $|x(t)| < .01$.

```
c = 2;
```

```
soln = DSolve[{x''[t] + c * x'[t] + x[t] == 0, x[0] == 1, x'[0] == 0}, x[t], t]
```

```
{{x[t] → e^-t (1 + t)}}
```

```
Plot[x[t] /. soln, {t, 5, 10}, PlotRange → {{5, 10}, {0, 0.04}}, PlotStyle → Thick]
```



Check your answer by using the **Solve** command as follows.

```
soln[[1, 1, 2]]
```

```
e^-t (1 + t)
```

```
Solve[soln[[1, 1, 2]] == .01, t]
```

```
{{t → -0.996308}, {t → 6.63835}}
```

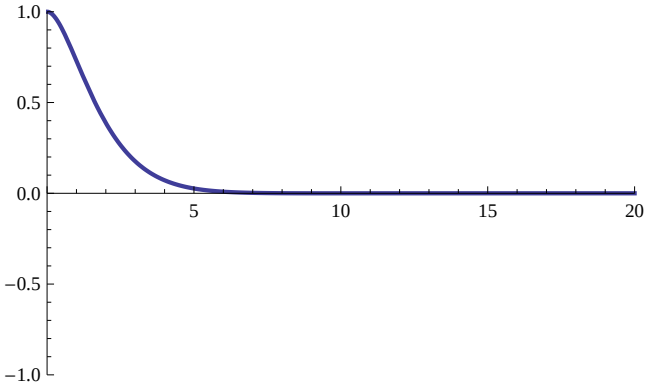
7. The value of $c = 2$ is the **critical damping value**. (See Nagle, Saff, and Snider, Section 4.9.) It is critical because if c were any smaller, oscillation will occur. Examine the solution and the graph of the solution in the case $c = 1.9$ for example to see if oscillation does occur.

```
c = 19 / 10;
```

```
soln = DSolve[{x''[t] + c * x'[t] + x[t] == 0, x[0] == 1, x'[0] == 0}, x[t], t]
```

$$\left\{\left\{x[t] \to \frac{1}{39}\, e^{-19\,t/20}\left(39 \operatorname{Cos}\left[\frac{\sqrt{39}\ t}{20}\right] + 19\sqrt{39}\ \operatorname{Sin}\left[\frac{\sqrt{39}\ t}{20}\right]\right)\right\}\right\}$$

```
Plot[x[t] /. soln, {t, 0, 20}, PlotRange → {{0, 20}, {-1, 1}}, PlotStyle → Thick]
```



If you did not see any oscillation, be sure you adjust the values of the second coordinate in the **PlotRange** option until the oscillation of the solution is visible on the graph. At approximately what value of *t* did you first see the graph begin to oscillate (rise again)? What is the approximate value of *x* at this value of *t*? The variable temp is the derivative of the function $x(t)$.

```
temp = D[soln[[1, 1, 2]], t]
```

$$\frac{1}{39}\, e^{-19\,t/20}\left(\frac{741}{20}\,\text{Cos}\left[\frac{\sqrt{39}\;t}{20}\right]-\frac{39}{20}\,\sqrt{39}\,\text{Sin}\left[\frac{\sqrt{39}\;t}{20}\right]\right)-$$

$$\frac{19}{780}\, e^{-19\,t/20}\left(39\,\text{Cos}\left[\frac{\sqrt{39}\;t}{20}\right]+19\,\sqrt{39}\,\text{Sin}\left[\frac{\sqrt{39}\;t}{20}\right]\right)$$

These commands will find a t value for which $x'(t) = 0$and will evaluate $x(t)$ at that point.

```
FindRoot[temp == 0, {t, 10}]
soln[[1, 1, 2]] /. FindRoot[temp == 0, {t, 10}]
```

```
{t → 10.0611}

-0.0000706275
```

You may want to check the graph of temp within a very narrow strip on the vertical axis to see what is going on here.

8. Try another positive value of c less than 2 that demonstrates more clearly from the graph that the solution is oscillatory.

9. By looking at the graphs of solutions of the damped oscillation equation for various values of *c*, find the value of *c* for which the time *tc* is minimal for which $|x(t)| < .01$for all $t > tc$.

Reference: Ben Pollina, "How Long Does It Take A Harmonic Oscillator to Come to Rest?", C-ODE-E, Summer-Fall 1995, pp 7-9.

Laboratory Nine

The Pendulum

(Section 4.8 of the Nagle/Saff/Snider text)

In this section we will model with second-order differential equations the motions of pendulums. The basic equation of the pendulum is

$$m\theta'' + k\theta' + \left(\frac{m\,g}{L}\right)\sin\theta = 0$$

where θ is the angle from the downward vertical measured positively in the counterclockwise direction, time is represented by *t*, *g* is the gravitational constant, *L* is the length of the pendulum, *k* is the coefficient of damping, and *m* is the mass. Clearly, the equation is nonlinear, which makes it an interesting equation to study since the analytic solution is not available.

Our objectives here are:

1. To see how *Mathematica* can handle nonlinear equations as this one with numerical procedures;

2. To determine the effect of the initial conditions for θ and its derivative on the motion of the pendulum as described by the differential equation model with and without damping;

3. To determine the effect of "linearizing" the differential equation; and

4. To determine the effect of the parameter *k* on the behavior of the pendulum.

Key *Mathematica* commands to be used in the section include **NDSolve** and **Plot**.

Let's first get our equation into the computer.

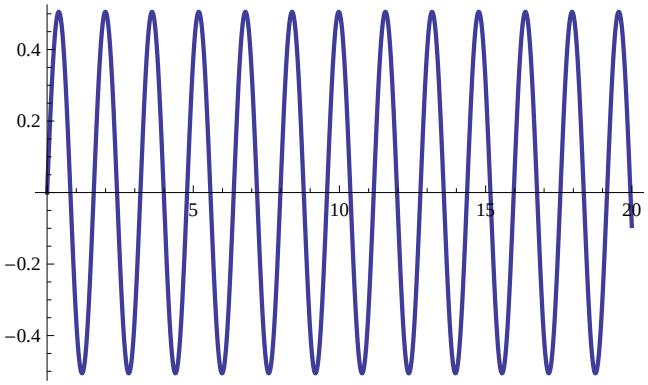
```
Clear[m, k, L, g];
eqn = m * D[θ[t], {t, 2}] + k * D[θ[t], t] + (m * g / L) * Sin[θ[t]] == 0
```

$$\frac{g\,m\,\text{Sin}[\theta[t]]}{L} + k\,\theta'[t] + m\,\theta''[t] == 0$$

9.1 The Undamped Case

Let us first assume that $k = 0$ and that we have a mass of 1 lb., a length of 2 feet, and that g = 32ft/sec^2.

```
k = 0; m = 1; g = 32; L = 2;
soln = NDSolve[{eqn, θ[0] == 0, θ'[0] == 2}, θ[t], {t, 0, 20}];
Plot[θ[t] /. soln, {t, 0, 20}, PlotStyle → Thick]
```

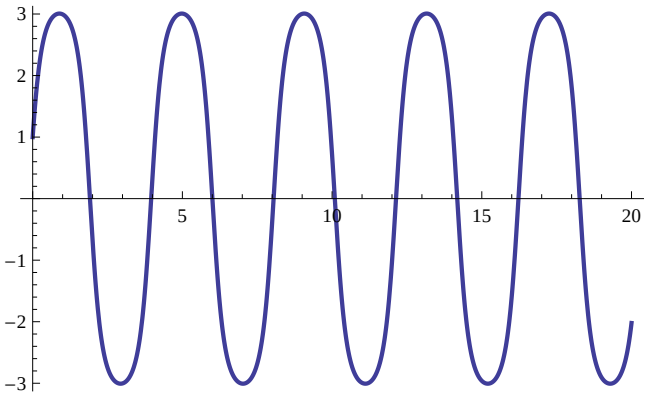


1. Look up **NDSolve** in the help menu and describe what the command does and how it does what it does.

2. Describe the behavior of the pendulum, in light of the initial conditions, by interpreting the graph above.

3. Now change the initial conditions by first using $\theta(0) = 1$ and $\theta'(0) = 7$ and then using $\theta(0) = 1$ and $\theta'(0) = 7.022$ and describe the behavior of the pendulum in each case. Decide whether the behavior makes sense according to the initial conditions. (You may need to make an adjustments in the **NDSolve** and **Plot** commands to change the range of θ.)

```
k = 0; m = 1; g = 32; L = 2;
soln = NDSolve[{eqn, θ[0] == 1, θ'[0] == 7}, θ[t], {t, 0, 20}];
Plot[θ[t] /. soln, {t, 0, 20}, PlotStyle → Thick]
```



4. Still working with the same command, let us determine the behavior if (a) $\theta(0)$ is an even or odd multiple of π or (b) $\theta(0)$ is very near an even or odd multiple of π and (c) $\theta'(0)$ is both small and large. Here are some variations to try. (Remember that π may be input as **Pi**.)

(a) $\theta(0) = \pi$ and $\theta'(0) = 0$

(b) $\theta(0) = 22/7$ and $\theta'(0) = 0$

(c) $\theta(0) = \pi$ and $\theta'(0) = 1$

(d) $\theta(0) = 2\pi$ and $\theta'(0) = 1$

(e) $\theta(0) = 10\pi$ and $\theta'(0) = 0$

(f) $\theta(0) = 220/7$ and $\theta'(0) = 0$

(g) $\theta(0) = -7\pi$ and $\theta'(0) = .04$

(h) $\theta(0) = -7\pi$ and $\theta'(0) = .01$

Comment on the behavior of the pendulum according to the graph in each case. Are any of the behaviors realistic? Are any of the behaviors not realistic? If so, what is the explanation for the unrealistic behavior? Do you see any ways to make improvements to get better results?

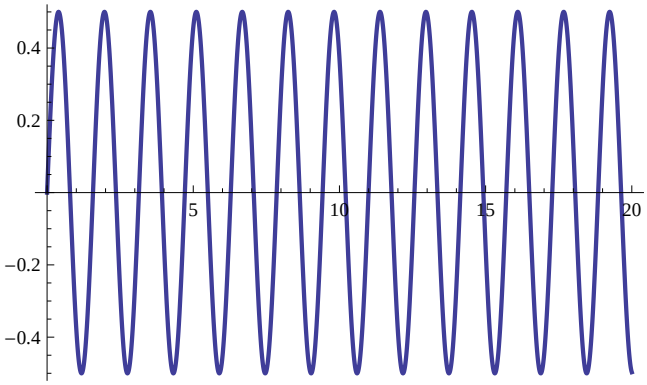
9.2 The Linearized Case

Let's now examine what happens when we linearize the equation of the pendulum by substituting θ for $\sin\theta$. We will still keep the damping zero for the present. The equation is now

$$m\theta'' + \left(\frac{m\,g}{L}\right)\theta = 0$$

```
Clear[m, k, L, g];
eqn = m * D[θ[t], {t, 2}] + k D[θ[t], t] + (m g) θ[t] / L == 0
k = 0; m = 1; g = 32; L = 2;
soln = NDSolve[{eqn, θ[0] == 0, θ'[0] == 2}, θ[t], {t, 0, 20}];
Plot[θ[t] /. soln, {t, 0, 20}, PlotStyle → Thick]
```

$$\frac{g\,m\,\theta[t]}{L} + k\,\theta'[t] + m\,\theta''[t] == 0$$



5. Experiment with the following initial values and summarize what happens.

(a) $\theta(0) = \pi$ and $\theta'(0) = 30$

(b) $\theta(0) = 1$ and $\theta'(0) = 7.022$

(c) $\theta(0) = 0$ and $\theta'(0) = 2$

(d) $\theta(0) = 2\pi$ and $\theta'(0) = 1$

(e) $\theta(0) = 10\pi$ and $\theta'(0) = 0$

For each case make a judgment whether the graph accurately describes the pendulum. You may need to refer to the "nonlinearized" cases above. Which of the graphs are fairly accurate and which are way off? Can you explain why?

9.3 With Damping Present

Now let us add a small damping to our original "non-linearized" equation and observe the effect.

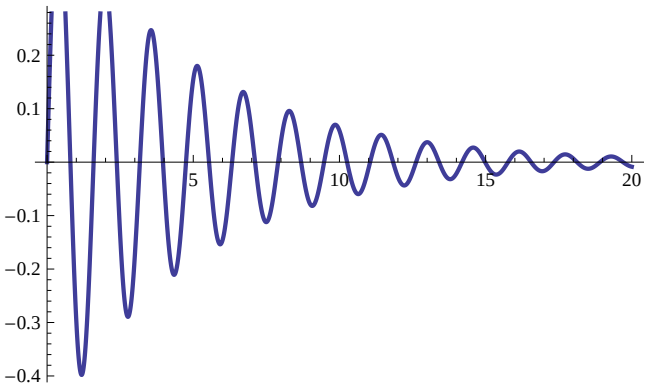
```
Clear[m, k, L, g];
eqn = m * D[θ[t], {t, 2}] + k * D[θ[t], t] + (m * g / L) * Sin[θ[t]] == 0
```

$$\frac{g\, m\, \text{Sin}[\theta[t]]}{L} + k\,\theta'[t] + m\,\theta''[t] == 0$$

6. Here are some cases to consider. Experiment with each and comment on the behavior of the pendulum.

(a) Take k=2, m=5, L=2, g = 32, $\theta(0) = 0$ and $\theta'(0) = 2$.

```
k = 2; m = 5; g = 32; L = 2;
soln = NDSolve[{eqn, θ[0] == 0, θ'[0] == 2}, θ[t], {t, 0, 20}];
Plot[θ[t] /. soln, {t, 0, 20}, PlotStyle → Thick]
```



(b) Take k = 2, m = 5, L = 2, g = 32, $\theta(0) = 1$ and $\theta'(0) = 1$.

(c) Take k = 2, m = 5, L = 2, g = 32, $\theta(0) = \pi$ and $\theta'(0) = 40$.

We will now look at the linearized equation that has a damping term present. For simplicity we will take m = 1 and L = 2. By animating the graphics we can determine the effect the damping coefficient has on the solution of the equation.

```
Clear[m, k, L, g];
plotlist = {};
eqn = D[θ[t], {t, 2}] + k * D[θ[t], t] + 16 * θ[t] == 0
soln = Table[DSolve[{eqn, θ[0] == π, θ'[0] == 0}, θ[t], t], {k, 4, 9, 0.2}];
For[n = 1, n ≤ Length[Flatten[soln]], n++, AppendTo[plotlist,
  Plot[θ[t] /. Flatten[soln][[n]], {t, 2, 5}, PlotRange → {-0.001, 0.001}, PlotStyle → Thick]]]
```

$$16\,\theta[t] + k\,\theta'[t] + \theta''[t] == 0$$

```
ListAnimate[plotlist]
```

7. By changing the domain of the values of "k" in the commands above, approximate the critical value of "k" where the solutions change from oscillating to non-oscillating. (You may need to change the range of t in order to see the graph better.) What should the answer be according to the theory? (See Section 4.8 of the text by Nagle, Saff, and Snider.)

8. **The Poe Pendulum.** Assume we have a pendulum with an ever increasing length over time. Assume the length at time t is given by $L = a + b\,t$ where a and b are constants. The linearized equation of the pendulum is then given as

$$(a + b\,t)\,\theta'' + 2\,b\,\theta' + m\,g\,\theta = 0$$

(See "Poe's Pendulum" by Borelli, Coleman, and Hobson in the *Mathematics Magazin*e, Vol. 58(1985) No. 2, pp. 78-83.) We are assuming that θ is small so that sin θ is approximately θ. The mass is 1.

(a) With *a* and *b* each equal to 1 and $\theta(0) = 1$ and $\theta'(0) = 0$, graph the solution. What happens to $\theta(t)$ as *t* gets large? Fill in the blanks marked **fill** with appropriate data. Since this is a numerical procedure, establishing a large interval for *t* could use considerable computer time.

```
Clear[m, k, L, g, a, b];
g = 32; m = 1; a = 1; b = 1;
eqn = (a + b t) D[θ[t], {t, 2}] + 2 b D[θ[t], t] + m g θ[t] == 0;
soln = NDSolve[{eqn, θ[0] == 1, θ'[0] == 0}, θ[t], {t, fill, fill}];
Plot[θ[t] /. soln, {t, fill, fill}, PlotStyle → Thick]
```

(b) As time gets larger, is the period of the solution fixed? What happens to the time of oscillation?

Laboratory Ten

Forced Equations and Resonance

(Section 4.10 of the Nagle/Saff/Snider text)

In this section we will take a deeper look at second-order nonhomogeneous equations. We will concentrate on equations that have a periodic forcing term. This will lead to a study of the phenomenon known as resonance.

The equations that we will consider will be of the following form:

$$m\,x'' + b\,x' + k\,x = \sin(\gamma\,t)$$

One can think of this as the equation of some spring - mass or a mechanical vibration that is being forced with a periodic function sin(γ t). Examples will be given below.

Our objectives here are the following.

1. To determine the effect of the forcing term on the behavior of the solution for different values of the parameter γ .

2. To understand the form of the solutions and the contributions of the various parts.

3. To understand resonance.

10.1 Undamped Case -- Exercises

Let's first take the equation m x''+k x=sin(γ t) in which the damping term involving *x'* is missing.

```
heqn = m * D[x[t], {t, 2}] + k * x[t] == 0 /. {m → 1, k → 3};
hsoln = DSolve[heqn, x[t], t]
```

1. Is this the expected solution of the homogeneous equation? What is the period and the frequency?

Let's add initial conditions to get a unique solution.

```
hsoln = DSolve[{heqn, x[0] == 1, x'[0] == 1}, x[t], t];
Plot[x[t] /. Flatten[hsoln], {t, 0, 10}, PlotStyle → Thick]
```

Does the graph confirm your answers?

Let's now add a forcing term, sin(γ t) to our equation.

```
feqn = m * D[x[t], {t, 2}] + k * x[t] == Sin[γ * t] /. {m → 1, k → 3};
fsoln = Simplify[DSolve[feqn, x[t], t]]
```

Is this the solution you expected?

2. Is the homogeneous solution still part of this solution? Write down the terms that are not part of the homogeneous solution. What values can γ not assume?

In order to examine the graph of the solution, let's again solve with initial conditions.

```
fsoln = Simplify[DSolve[{feqn, x[0] == 1, x'[0] == 1}, x[t], t]]
```

3. Now substitute some permissible values for γ into the solution and generate various graphs. Try $\gamma = 1, 1.5, 1.7, 1.73,$ and 2. Give a summary of what you observed. (In each case you must define the interval of the plot carefully to see the full effect.) Describe the long term behavior of the solution. Is there a dominating term in the solution that is responsible for most of the behavior? Fill in the blanks marked **fill** with appropriate data.

```
gammasoln = fsoln /. γ → fill
Plot[x[t] /. Flatten[gammasoln], {t, fill, fill}]
```

When $0 < b^2 < 4\,m\,k$, the general solution for m x''+b x'+k x=sin(γ t) is given by

$$x(t) = A\, e^{-\frac{bt}{2m}} \sin\left(\frac{\sqrt{4\,m\,k - b^2}}{2\,m}\, t + \phi\right) + \frac{1}{\sqrt{\left(k - m\,\gamma^2\right)^2 + b^2\,\gamma^2}} \sin(\gamma\, t + \theta)$$

where A, ϕ , and θ are constants. (See Section 4.9 of the text by Nagle, Saff, and Snider.) The first term represents the homogeneous part of the solution and the second represents the particular solution from the forcing term. When *b* = 0 and $0 < m\,k$, the solution becomes

$$x(t) = A \sin\left(\sqrt{\frac{k}{m}}\, t + \phi\right) + \frac{1}{k - m\,\gamma^2} \sin(\gamma\, t + \theta)$$

Currently, we are taking *b* = 0, *m* = 1, and *k* = 3 for various values of γ . Let's look more closely at the second term of the solution that comes from the forcing term. The coefficient of the sine function in this term is the *frequency gain function* and is the amplitude of the particular solution.

```
freqgain = 1 / Sqrt[(k - m * γ^2) ^2 + b^2 * γ^2];
psolnθ = freqgain * Sin[γ * t + θ];
```

4. Substitute the *b* = 0, *m* = 1, *k* = 3, and each of the γ values used above into the particular solution. Determine in each case the value of θ to match the portion of the solution **gammasoln** above that is the particular solution. Then graph the particular solution and the homogeneous solution on the same graph. Fill in the blanks marked **fill** with appropriate data.

```
psoln = psolnθ /. {b → 0, m → 1, k → 3, γ → fill, θ → fill}
Plot[{psoln, gammasoln[[1, 1, 2]] - psoln}, {t, 0, 10}, PlotStyle → {Red, Green}]
```

Now let's graph the frequency gain function as a function of γ.

```
freqgain /. {b → 0, m → 1, k → 3}
Plot[freqgain /. {b → 0, m → 1, k → 3}, {γ, 1, 2}]
```

5. At what value of γ does the amplitude of the particular solution blow up to infinity?

We have come very close to letting γ be the prohibited value. Now we will reconsider the problem by letting γ be this prohibited value. Fill in the blanks marked **fill** with appropriate data.

```
feqn = m D[x[t], {t, 2}] + k x[t] == Sin[γ t] /. {m → 1, k → 3, γ → fill};
fsoln = Simplify[DSolve[{feqn, x[0] == 1, x'[0] == 1}, x[t], t]];
Plot[x[t] /. Flatten[fsoln], {t, 0, fill}]
```

6. Describe the graph and the long term behavior of the solution. What term(s) in the solution contribute most to the long term behavior of the solution?

10.2 With Damping -- Exercises

Let's now add a damping term to our equation.

```
eqn = m * D[x[t], {t, 2}] + b * D[x[t], t] + k * x[t] == Sin[γ * t] /. {m → 1, b → 1 / 10, k → 2};
soln = Simplify[DSolve[{eqn, x[0] == 1, x'[0] == 0}, x[t], t]]
```

This solution looks very complicated but in reality consists of two parts -- a *steady-state* part that endures as *t* gets larger and a *transient* part that dies out as *t* gets larger.

7. Describe in a sentence the part of the solution above that is the steady-state part of the solution? Are there any values that γ cannot be?

Let's have *Mathematica* extract the steady-state portion of the solution. To do this we will remove the terms that have an exponential component. We first distribute to isolate the non-exponential terms, then subtract the exponential terms.

```
solnexp = Distribute[soln[[1, 1, 2]]]

steadystate = Simplify[solnexp - Coefficient[solnexp, Exp[-t / 20]] * Exp[-t / 20]]
```

Using trigonometry this can be written as $A\sin(\gamma t + \theta)$ (which is $A\sin(\gamma t)\cos\theta + A\cos(\gamma t)\sin\theta$) for some constant θ. Let's compute the square of the amplitude by selecting the coefficients of the cosine and sine terms, squaring each, and adding them together.

```
a1 = Coefficient[steadystate, Cos[tγ]]
a2 = Coefficient[steadystate, Sin[tγ]]
amplitude = Simplify[Sqrt[a1^2 + a2^2]]
```

8. How does this expression compare with the freqgain expression given earlier? By plotting the amplitude function, determine any values of gamma that give a maximum amplitude for the steady-state part of the function. By taking the derivative, determine any such values to within three decimal points. Find the maximum amplitude.

```
Plot[amplitude, {γ, -1, 2}]
crit = Solve[D[amplitude, γ] == 0, γ]
amplitude /. crit
```

9. Plot for various values of γ the graph of the solution. Can you pick out the graph of the transient part and the graph of the steady-state part? Suggested values of γ are 0, .1, 3, but also include the value that gives the maximum amplitude for the steady state solution. Fill in the blanks marked **fill** with appropriate data. Explain the result when γ = 0.

```
gammasoln = soln /. γ → fill

Plot[x[t] /. gammasoln, {t, 0, fill}]
```

If you had trouble picking out the transient and steady-state portions of the graph, we will let *Mathematica* assist us in picking out the steady-state and the transient solution for various values of γ. Fill in the blanks marked **fill** with appropriate data.

```
gammasoln = soln /. γ → fill;
gammasolnexp = Distribute[gammasoln[[1, 1, 2]]]
transient = Coefficient[gammasolnexp, e^(-t/20)] e^(-t/20)
steadystate = Simplify[gammasolnexp - transient]
Plot[{steadystate, transient}, {t, 0, fill}, PlotStyle → {Red, Green}]
```

10.3 Resonance -- Exercises

The frequency of the forcing term $\sin(\gamma t)$ is $\frac{\gamma}{2\pi}$. When we choose γ to be the value that gives the maximum amplitude of the steady-state solution, then we say we are at *resonance frequency*. When a mechanical system is stimulated by an external force that is at the resonance frequency, it is said to be at *resonance*.

The amplitude (or the frequency gain) is dependent on *m, k,* b, and γ . The next exercise treats it as a function of *b* and γ for fixed *m* and k.

$$\text{freqgain} = \frac{1}{\sqrt{\left(k - m\,\gamma^2\right)^2 + b^2\,\gamma^2}} /. \{m \to 1, k \to 3\};$$

```
plotlist = {}; For[n = 1, n ≤ 100, n++,
  AppendTo[plotlist, Plot[freqgain /. b → 1/n, {γ, 1, 3}, PlotRange → {0, 50}]]]
plotlist
```

10. Animate the above display and tell what happens to the critical numbers for the frequency gain function as *b* approaches zero. What is the value? How are the resonance frequencies of the damped, forced equations when *b* becomes small related to the frequency of the homogeneous equation without damping? What is happening to the maximum values of the frequency gain as *b* approaches 0?

11. If x''+b x' +3x=sin(γ t) represents the motion of a spring-mass system and γ is chosen so that we are at the resonance frequency, describe what happens physically to the system if the damping is small.

Laboratory Eleven

Laplace Transformations and Projectile Motion

(Chapter 7 and Section 3.4 of the Nagle/Saff/Snider text)

The Laplace transform is a very common and useful technique for solving and analyzing the solutions of initial-value problems involving linear and constant coefficient equations and systems. The key property of the Laplace transform is that derivatives are transformed into powers; thus, the differential equation is transformed into an algebraic equation. The focus of this notebook is to understand the Laplace transform and the inverse Laplace transform more than the solution of a differential equation.

Our objectives are as follows.

1. To understand the definition of the Laplace transform and the inverse Laplace transform.

2. To appreciate the general applicability of Laplace transforms to the solution of linear ODEs (both single equations and systems).

3. To understand and be able to work with piecewise-defined functions.

4. To become familiar with ways in which *Mathematica* can deal with Laplace transforms.

5. To use the Laplace transform to solve systems of equations that model the motion of a projectile.

Key *Mathematica* commands used in this section are: **`LaplaceTransform`**, **`InverseLaplaceTransfrom`**, and **`Apart`**.

11.1 Introduction to Laplace Transforms

The Laplace transform is defined in terms of an improper integral. For example,using the *Mathematica* command **`Integrate`** for integration, we can compute the Laplace transform of the three functions: 1, e^{at}, and cos(at) as follows.

```
Integrate[1 * Exp[-s * t], {t, 0, ∞}]
Integrate[Exp[a * t] * Exp[-s * t], {t, 0, ∞}]
Integrate[Cos[a * t] * Exp[-s * t], {t, 0, ∞}]
```

$$\text{If}\left[\text{Re}[s] > 0,\ \frac{1}{s},\ \text{Integrate}\left[e^{-s\,t},\ \{t,\ 0,\ \infty\},\ \text{Assumptions} \to \text{Re}[s] \le 0\right]\right]$$

$$\text{If}\left[\text{Re}[a] < \text{Re}[s],\ \frac{1}{-a+s},\ \text{Integrate}\left[e^{(a-s)\,t},\ \{t,\ 0,\ \infty\},\ \text{Assumptions} \to \text{Re}[a-s] \ge 0\right]\right]$$

$$\text{If}\left[a \in \text{Reals}\ \&\&\ \text{Re}[s] > 0,\ \frac{s}{a^2+s^2},\ \text{Integrate}\left[e^{-s\,t}\,\text{Cos}[a\,t],\ \{t,\ 0,\ \infty\},\ \text{Assumptions} \to a \notin \text{Reals}\ ||\ \text{Re}[s] \le 0\right]\right]$$

Notice that these are not the expected results. In each case, the anticipated result is present, but so are additional cases.

Questions to ponder:
- What do we know about *s* that *Mathematica* does not know?
- How does this knowledge allow us to compute the value of each integral?

The **LaplaceTransform** command does know the necessary information about the transform parameter. Here are the same problems using **LaplaceTransform** :

```
LaplaceTransform[1, t, s]
LaplaceTransform[Exp[a * t], t, s]
LaplaceTransform[Cos[a * t], t, s]
```

$$\frac{1}{s}$$

$$\frac{1}{-a+s}$$

$$\frac{s}{a^2+s^2}$$

Inverse Laplace transforms work similarly, except that there is no explicit formula. Regardless, it is still quite beneficial to use *Mathematica* to convert an expression into a form that can be inverted using the table of Laplace Transforms (inside back cover of the text by Nagle, Saff, and Snider). The **Apart** command is particularly helpful.

For example, the function

```
F1 = (7 s - 1) / (s^3 - 7 s - 6);
```

obviously has at least one real root. The partial fraction decomposition is

```
Apart[F1]
```

$$\frac{1}{-3+s}+\frac{2}{1+s}-\frac{3}{2+s}$$

The inverse Laplace transform is now easy to identify. **Check your answer** with the answer provided by *Mathematica:*

```
f1 = InverseLaplaceTransform[F1, s, t]
```

$$e^{-2t}\left(-3+2e^{t}+e^{5t}\right)$$

The next example

```
F2 = (s - 1) / (s^2 - 2 s + 5);
```

involves an irreducible quadratic term, which should be rewritten in a form with a complete square. This is achieved by using the **CompleteSquare** procedure:

```
CompleteSquare[p_, x_] :=
  Module[{a, b, c},
   {c, b, a} = CoefficientList[p, x];
   Return[a (x - (b / (2 a)))^2 + c - b^2 / 4 a]];

Numerator[F2] / CompleteSquare[Denominator[F2], s]
```

$$\frac{-1+s}{4+(1+s)^2}$$

The presence of (s-1) says that the inverse Laplace transform includes e^t. The inverse Laplace transform of $s/(s^2+4)$ is $\cos(2t)$; thus, the inverse Laplace transform of F2 must be $f(t) = e^t \cos(2t)$. As before, this answer is easily checked using *Mathematica*. Since *Mathematica* prefers to state the result in complex exponential form, we have to convert it to trigonometric form and simplify it.

```
f2 = FullSimplify[ExpToTrig[InverseLaplaceTransform[F2, s, t]]]
```

$$e^t \text{Cos}[2\,t]$$

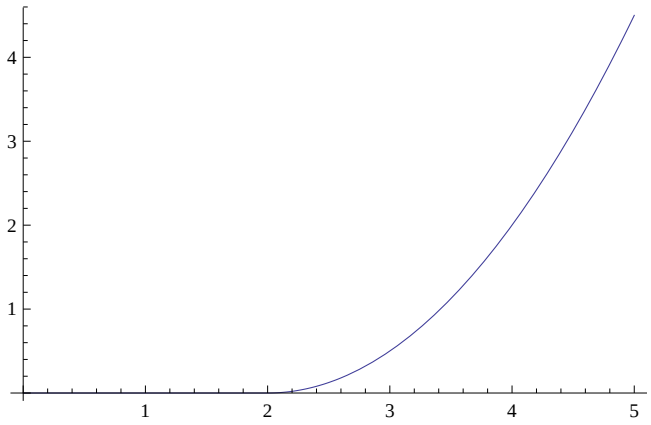
The third example looks innocuous enough, but the exponential term makes the inverse transform of this function interesting. We can still ask *Mathematica* for an answer.

```
F3 = Exp[-2 s] / s^3;
f3 = InverseLaplaceTransform[F3, s, t]
```

$$\frac{1}{2}(-2+t)^2 \,\text{HeavisideTheta}[-2+t]$$

What is this function `HeavisideTheta`? One of the easiest ways to learn about `HeavisideTheta` functions is to plot the function **f3**, and attempt to understand how it relates to the `HeavisideTheta` function.

```
Plot[f3, {t, 0, 5}]
```



Notice that the solution is identically zero for t<2 and is quadratic with a vertex at t=2 for t>2. This suggests that the `HeavisideTheta` function is 0 when its argument is negative and is identically 1 when the argument is positive.

11.2 Laplace Transforms for Initial Value Problems (Section 7.5 of the Nagle/Saff/Snider text)

Below are the problems that are of most interest.

Example 1. To illustrate the general procedure, consider the problem of solving the initial value problem

```
eqn = D[y[t], {t, 2}] - 3 * D[y[t], t] + 2 * y[t] == Cos[t] + (t^2 - 3) * Exp[-2 * t]
```

$$2\,y[t] - 3\,y'[t] + y''[t] == e^{-2t}\left(-3+t^2\right) + \text{Cos}[t]$$

using Laplace transforms.

The first step is to apply the Laplace transform to the entire equation. Since specific initial conditions are known, these should be introduced at the earliest convenience.

```
lapeqn = LaplaceTransform[eqn, t, s] /. {y[0] → 0, y'[0] → 1}
```

$$-1 + 2\,\text{LaplaceTransform}[y[t], t, s] - 3\,s\,\text{LaplaceTransform}[y[t], t, s] + s^2\,\text{LaplaceTransform}[y[t], t, s] == \frac{2}{(2+s)^3} - \frac{3}{2+s} + \frac{s}{1+s^2}$$

Note the way in which the Laplace transform of the solution is denoted in this equation. Next, it is necessary to explicitly solve for the Laplace transform of the solution:

```
solnlap = Solve[lapeqn, LaplaceTransform[y[t], t, s]][[1, 1, 2]]
```

$$\frac{-2 + 8\,s + 13\,s^2 + 7\,s^3 + 4\,s^4 + s^5}{(2+s)^3\,\left(1+s^2\right)\,\left(2-3\,s+s^2\right)}$$

The solution can be obtained in one step using **InverseLaplaceTransform**, but the solution is also available by inspection once the right-hand side is written in its partial fraction decomposition.

```
solnlap2 = Apart[solnlap]
```

$$\frac{109}{160\,(-2+s)} - \frac{31}{54\,(-1+s)} + \frac{1}{6\,(2+s)^3} + \frac{7}{72\,(2+s)^2} - \frac{179}{864\,(2+s)} + \frac{-3+s}{10\,\left(1+s^2\right)}$$

```
soln = InverseLaplaceTransform[solnlap2, s, t]
```

$$\frac{1}{4320}\,e^{-2t}\left(-895 - 2480\,e^{3t} + 2943\,e^{4t} + 420\,t + 360\,t^2 + 432\,e^{2t}\,\text{Cos}[t] - 1296\,e^{2t}\,\text{Sin}[t]\right)$$

Compare this solution with the solutions obtained by directly inverting the Laplace transform once it is available

```
InverseLaplaceTransform[solnlap, s, t]
```

$$\frac{1}{4320}\,e^{-2t}\left(-895 - 2480\,e^{3t} + 2943\,e^{4t} + 420\,t + 360\,t^2 + 432\,e^{2t}\,\text{Cos}[t] - 1296\,e^{2t}\,\text{Sin}[t]\right)$$

or the solution obtained by having *Mathematica* solve the problem using **DSolve**.

```
DSolve[{eqn, y[0] == 0, y'[0] == 1}, y[t], t]
```

$$\left\{\left\{y[t] \to \frac{1}{4320}\,e^{-2t}\left(-895 - 2480\,e^{3t} + 2943\,e^{4t} + 420\,t + 360\,t^2 + 432\,e^{2t}\,\text{Cos}[t] - 1296\,e^{2t}\,\text{Sin}[t]\right)\right\}\right\}$$

Example 2. For a second example consider the problem of finding the general solution to

```
eqn = D[y[t], {t, 2}] + 2 * D[y[t], t] == Cos[2 * t]
```

$$2\,y'[t] + y''[t] == \text{Cos}[2\,t]$$

The Laplace transform solution of this equation is

```
lapeqn = LaplaceTransform[eqn, t, s]
solnlap = Solve[lapeqn, LaplaceTransform[y[t], t, s]][[1, 1, 2]]
soln = InverseLaplaceTransform[solnlap, s, t]
```

$$s^2\,\text{LaplaceTransform}[y[t],t,s]+2\,(s\,\text{LaplaceTransform}[y[t],t,s]-y[0])-s\,y[0]-y'[0]==\frac{s}{4+s^2}$$

$$\frac{\frac{s}{4+s^2}+2\,y[0]+s\,y[0]+y'[0]}{2\,s+s^2}$$

$$\frac{1}{8}\,(-\text{Cos}[2\,t]+\text{Sin}[2\,t])+\frac{1}{8}\,e^{-2\,t}\,(1-4\,y'[0])+\frac{1}{2}\,(2\,y[0]+y'[0])$$

Note the explicit use of the initial conditions in this solution.

As before, there are several alternate methods to reach the same solution; it is particularly noteworthy to compare this solution with the one returned by **`DSolve`**:

```
DSolve[eqn, y[t], t]
```

$$\left\{\left\{y[t]\to-\frac{1}{2}\,e^{-2\,t}\,C[1]+C[2]-\frac{1}{8}\,\text{Cos}[2\,t]+\frac{1}{8}\,\text{Sin}[2\,t]\right\}\right\}$$

Example 3. The same techniques can be used to solve problems that involve piecewise-defined forcing functions (which are called Heaviside functions and which *Mathematica* calls `UnitStep`) and/or impulse functions (which are called Dirac delta functions and which *Mathematica* calls `DiracDelta`). For example, the general solution to the equation

```
eqn = D[y[t], {t, 2}] + 2 * D[y[t], t] == 3 * DiracDelta[t - 1] + UnitStep[t - π] * Sin[t - π] + Exp[-t]
```

$$2\,y'[t]+y''[t]==e^{-t}+3\,\text{DiracDelta}[-1+t]-\text{Sin}[t]\,\text{UnitStep}[-\pi+t]$$

can be found using the Laplace transform, exactly as before (this should illustrate the general structure of these problems):

```
lapeqn = LaplaceTransform[eqn, t, s]
solnlap = Solve[lapeqn, LaplaceTransform[y[t], t, s]][[1, 1, 2]]
soln = InverseLaplaceTransform[solnlap, s, t]
```

$$s^2\,\text{LaplaceTransform}[y[t],t,s]+2\,(s\,\text{LaplaceTransform}[y[t],t,s]-y[0])-s\,y[0]-y'[0]==3\,e^{-s}+\frac{1}{1+s}+\frac{e^{-\pi\,s}}{1+s^2}$$

$$\frac{3\,e^{-s}+\frac{1}{1+s}+\frac{e^{-\pi\,s}}{1+s^2}+2\,y[0]+s\,y[0]+y'[0]}{2\,s+s^2}$$

$$\frac{1}{10}\,e^{-2\,t}\left(15\left(-e^2+e^{2\,t}\right)\text{HeavisideTheta}[-1+t]+\text{HeavisideTheta}[-\pi+t]\left(-e^{2\,\pi}+5\,e^{2\,t}+4\,e^{2\,t}\,\text{Cos}[t]+2\,e^{2\,t}\,\text{Sin}[t]\right)+5\left(1-2\,e^t-y'[0]+e^{2\,t}\,(1+2\,y[0]+y'[0])\right)\right)$$

This solution may look needlessly complicated. Let's see if it can be simplified

```
FullSimplify[soln]
```

$$\frac{1}{10} e^{-2t}\left(15\left(-e^{2}+e^{2t}\right)\text{HeavisideTheta}[-1+t]+\text{HeavisideTheta}[-\pi+t]\left(-e^{2\pi}+e^{2t}(5+4\cos[t]+2\sin[t])\right)+10e^{t}(-1+\cosh[t](1+y[0])+\sinh[t](y[0]+y'[0]))\right)$$

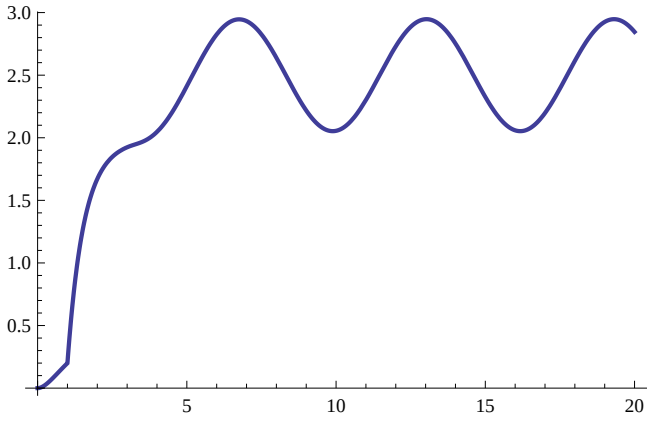
and written in a format that clearly identifies the role of the initial data and the different solutions on each interval

```
Collect[soln, {y[0], y'[0], HeavisideTheta[t - 1], HeavisideTheta[t - π]}]
```

$$\frac{1}{10} e^{-2t}\left(5-10e^{t}+5e^{2t}\right)+\frac{3}{2} e^{-2t}\left(-e^{2}+e^{2t}\right)\text{HeavisideTheta}[-1+t]+\frac{1}{10} e^{-2t}\text{HeavisideTheta}[-\pi+t]\left(-e^{2\pi}+5e^{2t}+4e^{2t}\cos[t]+2e^{2t}\sin[t]\right)+y[0]+\frac{1}{10} e^{-2t}\left(-5+5e^{2t}\right)y'[0]$$

A plot of the solution with initial conditions $y(0) = 0,\ y'(0) = 0$ is also illuminating.

```
Plot[soln /. {y[0] → 0, y'[0] → 0}, {t, 0, 20}, PlotStyle → Thick]
```



The same solution can be obtained directly using **DSolve**.

11.3 Laplace Transforms for Systems of Linear ODEs

One of the strengths of the Laplace transform is its applicability to both single equations and systems using exactly the same procedure; the only change is that now the algebraic equations will be a linear system. We illustrate with a single example.

```
sys = {D[x[t], t] == y[t] + Sin[t], D[y[t], t] == x[t] + Cos[t]}
lapeqn = LaplaceTransform[sys, t, s]
solnlap = Solve[lapeqn, {LaplaceTransform[x[t], t, s], LaplaceTransform[y[t], t, s]}]
soln = InverseLaplaceTransform[{solnlap[[1, 1, 2]], solnlap[[1, 2, 2]]}, s, t]
Collect[soln, {x[0], y[0]}]
```

$$\{x'[t] == \text{Sin}[t] + y[t],\ y'[t] == \text{Cos}[t] + x[t]\}$$

$$\left\{s\,\text{LaplaceTransform}[x[t], t, s] - x[0] == \frac{1}{1+s^2} + \text{LaplaceTransform}[y[t], t, s],\right.$$
$$\left. s\,\text{LaplaceTransform}[y[t], t, s] - y[0] == \frac{s}{1+s^2} + \text{LaplaceTransform}[x[t], t, s]\right\}$$

$$\left\{\left\{\text{LaplaceTransform}[x[t], t, s] \to -\frac{-2\,s - s\,x[0] - s^3\,x[0] - y[0] - s^2\,y[0]}{\left(-1+s^2\right)\left(1+s^2\right)},\right.\right.$$
$$\left.\left.\text{LaplaceTransform}[y[t], t, s] \to -\frac{-1 - x[0] - s\,y[0]}{-1+s^2}\right\}\right\}$$

$$\left\{-\text{Cos}[t] - \frac{1}{2}e^{t}\,(-1 - x[0] - y[0]) - \frac{1}{2}e^{-t}\,(-1 - x[0] + y[0]),\right.$$
$$\left. -\frac{1}{2}e^{t}\,(-1 - x[0] - y[0]) - \frac{1}{2}e^{-t}\,(1 + x[0] - y[0])\right\}$$

$$\left\{\frac{e^{-t}}{2} + \frac{e^{t}}{2} - \text{Cos}[t] + \left(\frac{e^{-t}}{2} + \frac{e^{t}}{2}\right)x[0] + \left(-\frac{e^{-t}}{2} + \frac{e^{t}}{2}\right)y[0],\right.$$
$$\left. -\frac{e^{-t}}{2} + \frac{e^{t}}{2} + \left(-\frac{e^{-t}}{2} + \frac{e^{t}}{2}\right)x[0] + \left(\frac{e^{-t}}{2} + \frac{e^{t}}{2}\right)y[0]\right\}$$

11.4 Exercises

1. Summarize the various ways of solving differential equations using the Laplace transform and *Mathematica* as shown above.

■ Exercises on Projectile Motion (Section 3.4 of the Nagle/Saff/Snider text)

2. A projectile shot from a gun has weight $w = m\,g$ and initial velocity v_0 tangent to its path of motion. Ignoring air resistance and all other forces except its weight, the system of differential equations describing the motion of the projectile is

$$m\frac{d^2x}{dt^2} = 0 \text{ and } m\frac{d^2y}{dt^2} = -m\,g.$$

Solve these equations subject to $x(0) = 0$, $y(0) = 0$, $x'(0) = v_0\cos\theta$, and $y'(0) = v_0\sin\theta$, where $v_0 = |v_0|$ is a constant and θ is the constant angle of elevation. Fill in blanks marked **`fill`** with appropriate data.

```
sys1 = {m * D[x[t], {t, 2}] == 0, m * D[y[t], {t, 2}] == -m * g};
lap1eqn =
 LaplaceTransform[sys1, t, s] /. {x[0] → 0, y[0] → 0, x'[0] → v0 * Cos[θ], y'[0] → v0 * Sin[θ]}
soln1lap = Solve[lap1eqn, {LaplaceTransform[x[t], t, s], LaplaceTransform[y[t], t, s]}]
soln1 = InverseLaplaceTransform[{soln1lap[[1, 1, 2]], soln1lap[[1, 2, 2]]}, s, t]
soln1x = soln1[[fill]]
soln1y = soln1[[fill]]
```

3. Use y(t) obtained in Exercise 2 to determine when the projectile lands. Use the expression $x(t)$ to show that the horizontal range of the projectile is given by $R = \frac{v_0^2}{g}\sin(2\theta)$. From this formula, we see that R is a maximum when $\sin(2\theta) = 1$, or $\theta = \frac{\pi}{4}$. Fill in blanks marked **fill** with appropriate data.

```
T1 = Solve[soln1y == 0, t]

HorRan1 = fill /. T1[[fill]]

Simplify[HorRan1]
```

4. Now suppose that air resistance is a retarding force tangent to the path but acts opposite to the motion. Since air resistance is then a multiple of the velocity, the system of differential equations describing the motion of the projectile is

$$m\frac{d^2x}{dt^2} = -\beta\frac{dx}{dt} \text{ and } m\frac{d^2y}{dt^2} = -mg - \beta\frac{dy}{dt}, \text{ where } \beta > 0.$$

Use the Laplace transform to solve these equations subject to $x(0) = 0$, $y(0) = 0$, $x'(0) = v_0\cos\theta$, and $y'(0) = v_0\sin\theta$. Fill in blanks marked **fill** with appropriate data.

```
sys2 = {m * D[x[t], {t, 2}] == -β * fill, m * D[y[t], {t, 2}] == -m * g - β * fill};
lap2eqn =
 LaplaceTransform[sys2, t, s] /. {x[0] → 0, y[0] → 0, x'[0] → v0 * Cos[θ], y'[0] → v0 * Sin[θ]}
soln2lap = Solve[lap2eqn, {LaplaceTransform[x[t], t, s], LaplaceTransform[y[t], t, s]}]
soln2 = InverseLaplaceTransform[{soln2lap[[1, 1, 2]], soln2lap[[1, 2, 2]]}, s, t]
soln2x = soln2[[fill]]
soln2y = soln2[[fill]]
```

5. Suppose m = 1/4 slug, g = 32 ft./sec^2, β = 0.02, θ = 38°, and v_0 = 300 ft/sec. Find the time when the projectile hits the ground and then compute the corresponding horizontal range. (Put θ in radians at the point marked **fill**.)

```
soln3y = soln2y /. {m → 1 / 4, g → 32, β → .02, θ → fill, v0 → 300}
T3 = FindRoot[soln3y, {t, 10}]
soln3x = soln2x /. {m → 1 / 4, g → 32, β → .02, θ → fill, v0 → 300}
HorRan3 = soln3x /. fill
```

6. Use the equations $x(t)$ and $y(t)$ in Exercise 2 with θ = 38°, and v_0 = 300 ft/sec. and the analogous equations in part (5) as parametric equations to plot the path of the projectile -- the ballistic curve -- on the same coordinate system. Give a time interval [0,T] , where T is the time the projectile hits the ground. Fill in blanks marked **fill** with appropriate data.

```
soln4x = soln1x /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
soln4y = soln1y /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
T4 = T1 /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
HorRan4 = HorRan1 /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
g1 = ParametricPlot[{soln4x, soln4y}, {t, 0, t /. fill}, PlotStyle → {Thick, Red}];
g2 = ParametricPlot[{soln3x, soln3y}, {t, 0, t /. fill}, PlotStyle → {Thick, Blue}];
Show[g1, g2,
 PlotLabel → "Projectiles Fired at 38° with (blue) and without (red) air resistance"]
```

7. For the equations in Exercise 2, also plot the ballistic curve for the complementany angle θ = 52°. In Exercise 5, show that the complementary angle does not give the same range. Also plot this ballistic curve. Fill in blanks marked **fill** with appropriate data.

```
soln5x = soln1x /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
soln5y = soln1y /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
T5 = T1 /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
HorRan5 = HorRan1 /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
soln6x = soln2x /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
soln6y = soln2y /. {m → 1/4, g → 32, β → 0.02, θ → fill, v0 → 300}
T6 = FindRoot[soln6y, {t, fill}]
HorRan6 = soln3x /. T6
g3 = ParametricPlot[{soln5x, soln5y}, {t, 0, t /. fill}, PlotStyle → {Thick, Red}];
g4 = ParametricPlot[{soln6x, soln6y}, {t, 0, t /. fill}, PlotStyle → {Thickn, Blue}];
Show[g3, g4,
 PlotLabel → "Projectiles Fired at 52° with (blue) and without (red) air resistance"]
```

Write your conclusion from the graphs.

8. A motorcyclist wants to jump over the Grand Canyon on a motorcycle. Assuming that his body weight and the weight of the motorcycle together are 500 pounds and the takeoff ramp is at 38° from the horizontal, how far can he be expected to jump if the initial velocity on take off is 90 miles per hour? Assume that there is a wind along the direction of motion so that β has a value of .1. (1 slug is equivalent to approximately 32.2 pounds. 1 mile is 5280 feet.)

Laboratory Twelve

Analytical and Graphical Analysis of Systems

(Sections 5.1, 5.2, 5.4 and 9.1-9.6 of the Nagle/Saff/Snider text)

In this section we will look at two-dimensional linear systems with the help of *Mathematica*. These systems are of the form

$$\frac{dx}{dt} = ax + by$$
$$\frac{dy}{dt} = cx + dy$$

where *a, b, c*, and *d* are real constants. We will write this in matrix form as

$$\frac{dY}{dt} = AY$$

where *A* is the matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and *Y* is the matrix $\begin{pmatrix} x \\ y \end{pmatrix}$.

Our objectives in this section are as follows.

1. To learn how to use *Mathematica* to find the eigenvalues, eigenvectors, and the general solution of a two-dimensional system.

2. To learn how to use *Mathematica* to find unique solutions to initial-value problems involving two-dimensional systems.

3. To learn how to use *Mathematica* to examine the phase plane of two-dimensional systems.

Key *Mathematica* commands used in this section are **`Eigensystem`**, **`Eigenvalues`**, **`Eigenvectors`**, and **`Plot`**.

12.1 An Example (Section 9.5 of the Nagle/Saff/Snider text)

Let us solve the two-dimensional system given by the following equation: $\frac{dY}{dt} = \begin{pmatrix} 1 & 4 \\ 2 & -1 \end{pmatrix} Y$. To compute the eigenvalues and eigenvectors we use the **`Eigensystem`** command.

```
A = {{1, 4}, {2, -1}}
ev = Eigensystem[A]
```

```
{{1, 4}, {2, -1}}
```

```
{{-3, 3}, {{-1, 1}, {2, 1}}}
```

This last result gives us a list of eigenvalues and another list whose entries are the corresponding eigenvectors. Thus -3 is an eigenvalue with corresponding eigenvector [-1,1], and the other eigenvalue is 3 with eigenvector [2,1]. Alternatively we could find the eigenvalues separately by

```
λ = Eigenvalues[A]
```

```
{-3, 3}
```

Now we can write the general solution of the system as

```
Y = k1 * Exp[λ[[1]] * t] * {-1, 1} + k2 * Exp[λ[[2]] * t] * {2, 1}
```

```
{-ℯ^(-3 t) k1 + 2 ℯ^(3 t) k2, ℯ^(-3 t) k1 + ℯ^(3 t) k2}
```

If in addition we are given an initial value for Y at $t = 0$, say $Y(0) = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$, then let's find out what $k1$ and $k2$ must be in order for this to be the initial value of the solution. Let's first name the initial value **ival**:

```
ival = {1, 4}
(Y /. t → 0) == ival
```

```
{1, 4}
```

```
{-k1 + 2 k2, k1 + k2} == {1, 4}
```

```
M = (Y /. t → 0) - ival
```

```
{-1 - k1 + 2 k2, -4 + k1 + k2}
```

Now we solve for the constants $k1$ and $k2$ in row 1 and 2 of the matrix M.

```
kvals = Solve[M == {0, 0}, {k1, k2}]
```

$$\left\{\left\{k1 \to \frac{7}{3},\ k2 \to \frac{5}{3}\right\}\right\}$$

```
solnY = Y /. Flatten[kvals]
```

$$\left\{-\frac{7}{3}\,ℯ^{-3\,t} + \frac{10\,ℯ^{3\,t}}{3},\ \frac{7\,ℯ^{-3\,t}}{3} + \frac{5\,ℯ^{3\,t}}{3}\right\}$$

Here is a check that we have actually attained the solution.

```
Simplify[D[solnY, t] - A.solnY]
```

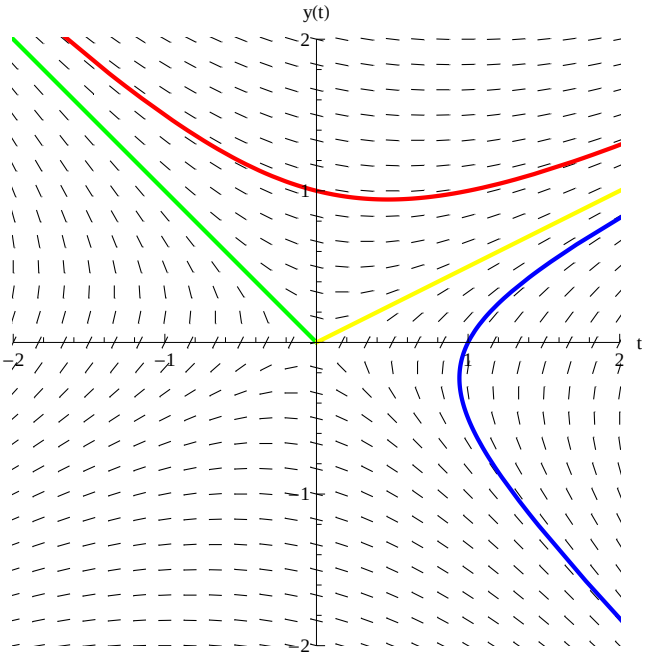
```
{0, 0}
```

This last command tells *Mathematica* to return the matrix obtained by subtracting the product A.solnY from the matrix obtained by differentiating each component of solnY with respect to t.

Let us now also get a graphical picture of solutions to the system by examining the phase plane. Since we have two distinct real eigenvalues of opposite sign, we know that there should be two straight line solutions with one tending away from the origin (the one on the vector [2,1]) and the other tending toward the origin (the one on the vector [-1,1]) as t goes to ∞. In this situation the origin is called a *saddle*. All the other solutions will be governed by the behavior of these two solutions as the general solution Y above indicates.

To generate the phase plane we will write down the two equations for the system and use the **VectorPlot** and **ParametricPlot** commands.

```
dir = VectorPlot[{x + 4 y, 2 x - y}, {x, -2, 2}, {y, -2, 2}, VectorPoints → 25, Axes → True,
    Frame → False, VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.015, 1, None},
    PlotRange → {{-2, 2}, {-2, 2}}, AxesOrigin → {0, 0}, AxesLabel → {"t", "y(t)"}];
soln1 = Flatten[Solve[(Y /. t → 0) == {1, 1}, {k1, k2}]];
g1 = ParametricPlot[Y /. soln1, {t, -2, 2}, PlotStyle → {Thick, Red}];
soln2 = Flatten[Solve[(Y /. t → 0) == {-1, 1}, {k1, k2}]];
g2 = ParametricPlot[Y /. soln2, {t, -2, 2}, PlotStyle → {Thick, Green}];
soln3 = Flatten[Solve[(Y /. t → 0) == {2, 1}, {k1, k2}]];
g3 = ParametricPlot[Y /. soln3, {t, -2, 2}, PlotStyle → {Thick, Yellow}];
soln4 = Flatten[Solve[(Y /. t → 0) == {1, 0}, {k1, k2}]];
g4 = ParametricPlot[Y /. soln4, {t, -2, 2}, PlotStyle → {Thick, Blue}];
Show[dir, g1, g2, g3, g4, PlotRange → {{-2, 2}, {-2, 2}}]
```



From the phase plane generated together with the four solutions through the given initial points, the two straight line solutions are clearly visible, one approaching the origin and the other leaving the origin. Also, the behavior of the other solutions is visible, coming from -∞ along one straight line solution and going to ∞ along the other.

12.2 Exercises

1. Find a two-dimensional linear system of differential equations that has two distinct positive eigenvalues. Find the eigenvectors, construct the solution, and generate the phase plane. Then summarize the essential features of the phase plane. You will probably need commands like these below. Fill in the blanks marked **`fill`** with appropriate data.

```
Clear[k1, k2];
A = {{fill, fill}, {fill, fill}};
ev = Eigensystem[A];
λ = Eigenvalues[A];
Y = k1 e^(λ[[1]] t) {fill, fill} + k2 e^(λ[[2]] t) {fill, fill};
dir = VectorPlot[{fill, fill}, {x, -2, 2}, {y, -2, 2}, VectorPoints → 25, Axes → True,
   Frame → False, VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.015, 1, None},
   PlotRange → {{-2, 2}, {-2, 2}}, AxesOrigin → {0, 0}, AxesLabel → {"t", "y(t)"}];
soln1 = Flatten[Solve[(Y /. t → 0) == {fill, fill}, {k1, k2}]];
g1 = ParametricPlot[Y /. soln1, {t, -2, 2}, PlotStyle → {Thickness[0.006], Red}];
soln2 = Flatten[Solve[(Y /. t → 0) == {fill, fill}, {k1, k2}]];
g2 = ParametricPlot[Y /. soln2, {t, -2, 2}, PlotStyle → {Thickness[0.006], Green}];
soln3 = Flatten[Solve[(Y /. t → 0) == {fill, fill}, {k1, k2}]];
g3 = ParametricPlot[Y /. soln3, {t, -2, 2}, PlotStyle → {Thickness[0.006], Yellow}];
soln4 = Flatten[Solve[(Y /. t → 0) == {fill, fill}, {k1, k2}]];
g4 = ParametricPlot[Y /. soln4, {t, -2, 2}, PlotStyle → {Thickness[0.006], Blue}];
Show[dir, g1, g2, g3, g4, PlotRange → {{-2, 2}, {-2, 2}}]
```

2. Find a two-dimensional linear system of differential equations that has two distinct negative eigenvalues. Find the eigenvectors, construct the solution, and generate the phase plane. Then summarize the essential features of the phase plane. Adjust the commands used above to fit this situation.

3. If a two-dimensional system has only one real eigenvalue of multiplicity two, it is possible that the eigenvalue has two linearly independent eigenvectors. Consider for example the system $\frac{dY}{dt} = c\,I\,Y$ where c is a real number and I is the identity matrix. Pick a value for c, compute the eigenvalues, two linearly independent eigenvectors, the solution, and the phase plane. Summarize the essential features of the solutions from the phase plane.

In some two-dimensional linear systems, there may only be one eigenvalue λ and only one linearly independent eigenvector v. In this case we must find another vector u, called a *generalized eigenvector*, satisfying the equation $(A - \lambda I)\,u = v$. We can then form two solutions $e^{\lambda t}\,v$ and $e^{\lambda t}(t\,v + u)$.

4. Find a system that has only one real eigenvalue with only one linearly independent eigenvector. Generate an eigenvector, a generalized eigenvector, the general solution, and the phase plane. Summarize the essential features of the solutions from the phase plane. (To find the generalized eigenvector, use the following commands.)

```
A = {{fill, fill}, {fill, fill}};
ev = Eigenvectors[A]
λ = Eigenvalues[A]
B = A - λ[[1]] * IdentityMatrix[2]
u = LinearSolve[B, ev[[1]]]
```

In the previous commands we are computing the matrix $B = A - \lambda I$. The command **IdentityMatrix[n]** produces the $n \times n$ identity matrix, 2×2 in this case. We are assuming there is only one eigenvalue λ in this case. The LinearSolve command will solve the system $B\,u = v$ for u. If there are parameters in the solution, you can select them arbitrarily to find a suitable u.

```
Clear[k1, k2];
Y = k1 e^λ[[1]] t {fill, fill} + k2 e^λ[[2]] t (t {fill, fill} + {fill, fill});
dir = VectorPlot[{fill, fill}, {x, -2, 2}, {y, -2, 2}, VectorPoints → 25, Axes → True,
    Frame → False, VectorStyle → {Black, Arrowheads[0]}, VectorScale → {.015, 1, None},
    PlotRange → {{-2, 2}, {-2, 2}}, AxesOrigin → {0, 0}, AxesLabel → {"t", "y(t)"}];
soln1 = Flatten[Solve[(Y /. t → 0) == {fill, fill}, {k1, k2}]];
g1 = ParametricPlot[Y /. soln1, {t, -2, 2}, PlotStyle → {Thickness[0.006], Red}];
soln2 = Flatten[Solve[(Y /. t → 0) == {fill, fill}, {k1, k2}]];
g2 = ParametricPlot[Y /. soln2, {t, -2, 2}, PlotStyle → {Thickness[0.006], Green}];
soln3 = Flatten[Solve[(Y /. t → 0) == {fill, fill}, {k1, k2}]];
g3 = ParametricPlot[Y /. soln3, {t, -2, 2}, PlotStyle → {Thickness[0.006], Yellow}];
soln4 = Flatten[Solve[(Y /. t → 0) == {fill, fill}, {k1, k2}]];
g4 = ParametricPlot[Y /. soln4, {t, -2, 2}, PlotStyle → {Thickness[0.006], Blue}];
Show[dir, g1, g2, g3, g4, PlotRange → {{-2, 2}, {-2, 2}}]
```

12.3 Complex Eigenvalues (Section 9.6 of the Nagle/Saff/Snider text)

In the situation where an eigenvalue λ is a complex number, a solution is still given by $e^{\lambda t} u$, where u is a corresponding eigenvector. However in this case both the real and the imaginary parts of this solution are real-valued solutions. We must therefore extract the real and the imaginary parts in order to get the general solution. Let's try an example.

```
A = {{-1, 6}, {-4, -6}};
ev = Eigenvectors[A]
λ = Eigenvalues[A]
```

$$\left\{\left\{-\frac{3}{2}+\frac{1}{8}\left(7-i\sqrt{71}\right),\ 1\right\},\ \left\{-\frac{3}{2}+\frac{1}{8}\left(7+i\sqrt{71}\right),\ 1\right\}\right\}$$

$$\left\{\frac{1}{2}\left(-7+i\sqrt{71}\right),\ \frac{1}{2}\left(-7-i\sqrt{71}\right)\right\}$$

The **ComplexExpand** command assumes that all variables involved are real, thus t is real in this case.

```
tempsoln = ComplexExpand[ComplexExpand[Exp[t * λ[[1]]]] * ComplexExpand[ev[[1]]]]
```

$$\left\{-\frac{5}{8}e^{-4t}\sqrt{e^t}\ \text{Cos}\left[\frac{\sqrt{71}\ t}{2}\right]+\frac{1}{8}\sqrt{71}\ e^{-4t}\sqrt{e^t}\ \text{Sin}\left[\frac{\sqrt{71}\ t}{2}\right]+\right.$$
$$i\left(-\frac{1}{8}\sqrt{71}\ e^{-4t}\sqrt{e^t}\ \text{Cos}\left[\frac{\sqrt{71}\ t}{2}\right]-\frac{5}{8}e^{-4t}\sqrt{e^t}\ \text{Sin}\left[\frac{\sqrt{71}\ t}{2}\right]\right),$$
$$\left.e^{-4t}\sqrt{e^t}\ \text{Cos}\left[\frac{\sqrt{71}\ t}{2}\right]+i\,e^{-4t}\sqrt{e^t}\ \text{Sin}\left[\frac{\sqrt{71}\ t}{2}\right]\right\}$$

```
soln1 = ComplexExpand[Re[tempsoln]]
```

$$\left\{-\frac{5}{8}e^{-4t}\sqrt{e^t}\ \text{Cos}\left[\frac{\sqrt{71}\ t}{2}\right]+\frac{1}{8}\sqrt{71}\ e^{-4t}\sqrt{e^t}\ \text{Sin}\left[\frac{\sqrt{71}\ t}{2}\right],\ e^{-4t}\sqrt{e^t}\ \text{Cos}\left[\frac{\sqrt{71}\ t}{2}\right]\right\}$$

```
soln2 = ComplexExpand[Im[tempsoln]]
```

$$\left\{-\frac{1}{8}\sqrt{71}\ e^{-4t}\sqrt{e^t}\ \text{Cos}\left[\frac{\sqrt{71}\ t}{2}\right]-\frac{5}{8}e^{-4t}\sqrt{e^t}\ \text{Sin}\left[\frac{\sqrt{71}\ t}{2}\right],\ e^{-4t}\sqrt{e^t}\ \text{Sin}\left[\frac{\sqrt{71}\ t}{2}\right]\right\}$$

```
Clear[k1, k2];
Y = k1 * soln1 + k2 * soln2;
```

Now let's find the solution with initial value at t = 0 as $\begin{pmatrix} 4 \\ 3 \end{pmatrix}$.

```
ival = {4, 3};
kvals = Flatten[Solve[(Y /. t → 0) == ival, {k1, k2}]]
solnY = Simplify[Y /. Flatten[kvals]]
```

$$\left\{k2 \to -\frac{47}{\sqrt{71}},\ k1 \to 3\right\}$$

$$\left\{\frac{4}{71}\, e^{-4\,t} \sqrt{e^{t}} \left(71 \operatorname{Cos}\left[\frac{\sqrt{71}\ t}{2}\right] + 14\sqrt{71}\ \operatorname{Sin}\left[\frac{\sqrt{71}\ t}{2}\right]\right),\right.$$

$$\left.\frac{1}{71}\, e^{-4\,t} \sqrt{e^{t}} \left(213 \operatorname{Cos}\left[\frac{\sqrt{71}\ t}{2}\right] - 47\sqrt{71}\ \operatorname{Sin}\left[\frac{\sqrt{71}\ t}{2}\right]\right)\right\}$$

12.4 Exercises

5. Use *Mathematica* to compute the eigenvalues, eigenvectors, the general solution, the solution with initial value $Y(0) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$, and the phase plane of the system $\frac{dY}{dt} = \begin{pmatrix} 2 & -6 \\ 2 & 1 \end{pmatrix} Y$. Is the origin a sink, a source, or a center?

6. Consider the following second-order equation $\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + 2x = 0$. Convert this into a two-dimensional system and use *Mathematica* to compute the eigenvalues, the eigenvectors, the general solution, and the phase plane. What happens to the solutions as $t \to \infty$?

7. Consider the three-dimensional system $\frac{dY}{dt} = \begin{pmatrix} -2 & 1 & 0 \\ -1 & 0 & 2 \\ -1 & 1 & 0 \end{pmatrix} Y$. Use *Mathematica* to find the eigenvalues, the eigenvectors, and the general solution. Find the solution that has initial value $Y(0) = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$.

8. Consider the three-dimensional system $\frac{dY}{dt} = \begin{pmatrix} -2 & 3 & 0 \\ -1 & 0 & 2 \\ -1 & 1 & 0 \end{pmatrix} Y$. Use *Mathematica* to find the eigenvalues, the eigenvectors, and the general solution.

9. In this exercise, we will solve the system of #7, without seeing the eigenvalues and the eigenvectors but let *Mathematica* generate the solution using the **DSolve** command. This approach can be tried in any system but the insight given by the eigenvectors and eigenvalues is missing. In terms of three equations, the system of #7 can be written as follows.

```
eqn1 = x'[t] == -2 x[t] + y[t];
eqn2 = y'[t] == -x[t] + 2 z[t];
eqn3 = z'[t] == -x[t] + y[t];
soln = DSolve[{eqn1, eqn2, eqn3}, {x[t], y[t], z[t]}, t]
solninit = DSolve[{eqn1, eqn2, eqn3, x[0] == 1, y[0] == 1, z[0] == 2}, {x[t], y[t], z[t]}, t]
```

How does the solution you obtained compare with the solution of #7? Compute the value of the solution at t = 2 with the following command.

```
solninit /. t → 2
```

How does this value compare with the value of the solution of #7 at the point t = 2?

Laboratory Thirteen

Numerical Techniques on Systems

(Sections 3.6, 3.7 and 5.3 of the Nagle/Saff/Snider text)

The harmonic oscillator is modeled by the equation

$$m \frac{d^2 y}{d t^2} + k_d \frac{d y}{d t} + k_s y = 0$$

where m is the mass, k_d is the coefficient of damping, and k_s is the spring constant. It can be written as a linear autonomous system of first-order equations by letting $v = \frac{dy}{dt}$ as follows:

$$\begin{aligned} \frac{dy}{dt} &= v \\ \frac{dy}{dt} &= -k_s y - k_d v \end{aligned}$$

Not only can this second-order equation be converted into a system of first-order equations but any m^{th} order differential equation can be converted into a system of m first-order equations. Moreover, most numerical procedures for approximating the solution to an initial-value problem for a higher-order differential equation require that the equation be converted first into a system of first-order equations. In this section we will look at approximating solutions to first-order systems by numerical techniques. We will in effect be looking at numerical solutions also of higher order equations. For simplicity we will limit ourselves to systems of two first-order equations as above. They are of the form

$$\begin{aligned} \frac{dx}{dt} &= f(t, x, y) \\ \frac{dy}{dt} &= g(t, x, y) \end{aligned}$$

Our objectives in this section are as follows.

1. To use Euler's method for systems to analyze solutions in the phase plane, the *t-x* plane, and the *t-y* plane.

2. To use other numerical methods to analyze solutions and compare results with Euler's method.

3. To look at the harmonic oscillator as a particular example and see the effect of parameters on the solutions.

4. To see the effect of changing the damping term of the harmonic oscillator to a nonlinear term.

13.1 Euler's Method on Systems

Let's begin with an example of a two-dimensional linear system.

```
diffeq1 = D[x[t], t] == 3 * x[t] - y[t]
diffeq2 = D[y[t], t] == x[t] + y[t]
```

```
x'[t] == 3 x[t] - y[t]
```

```
y'[t] == x[t] + y[t]
```

We will now use the **Dsolve** command to get a solution.

```
soln = DSolve[{diffeq1, diffeq2}, {x[t], y[t]}, t]
```

$$\left\{\left\{x[t] \to e^{2t}\,(1+t)\,C[1] - e^{2t}\,t\,C[2],\ y[t] \to e^{2t}\,t\,C[1] - e^{2t}\,(-1+t)\,C[2]\right\}\right\}$$

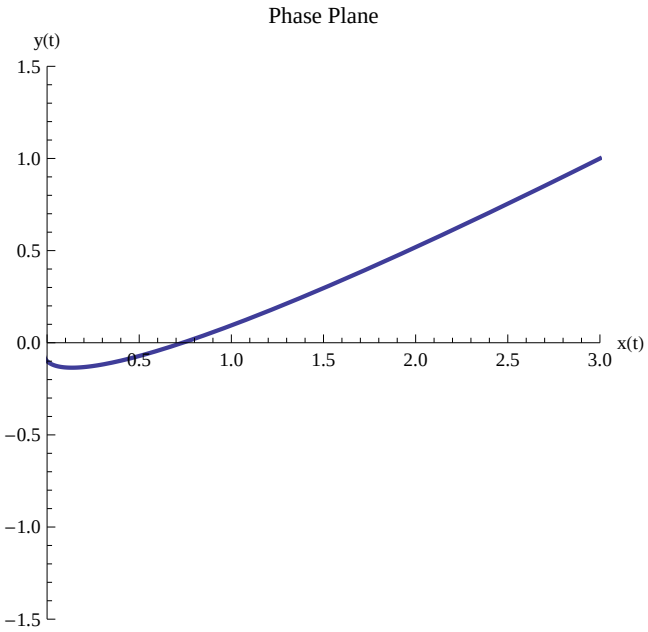
Notice that the the solutions are given in terms of two parameters `C[1]` and `C[2]`. Let's now introduce initial conditions.

```
soln = DSolve[{diffeq1, diffeq2, x[0] == 3, y[0] == 1}, {x[t], y[t]}, t]
```

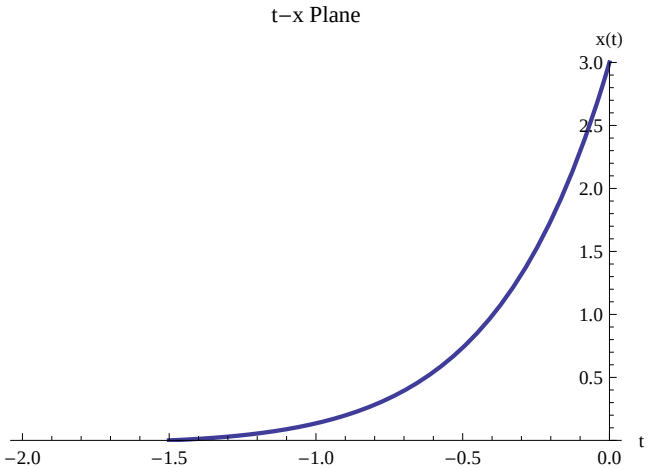
$$\left\{\left\{x[t] \to e^{2t}\,(3+2\,t),\ y[t] \to e^{2t}\,(1+2\,t)\right\}\right\}$$

We can now plot the solution in the phase plane, the *t-x* plane, and the *t-y* plane.

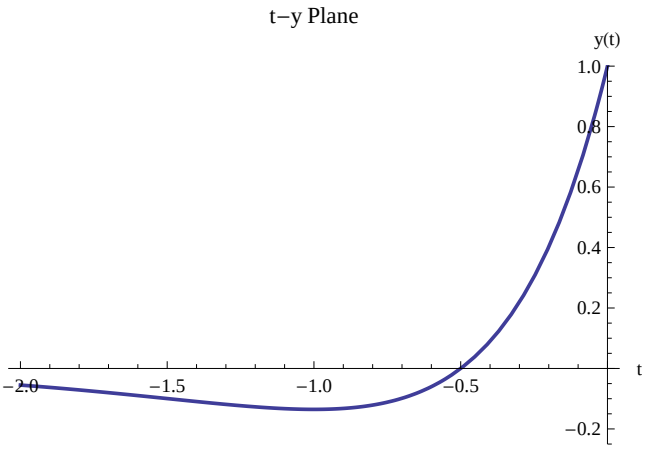
```
ParametricPlot[{(x[t] /. soln)[[1]], (y[t] /. soln)[[1]]},
 {t, -2, 0}, PlotRange → {{0, 3}, {-1.5, 1.5}}, PlotStyle → Thick,
 PlotLabel → "Phase Plane", AxesLabel → {"x(t)", "y(t)"}]
```




```
Plot[(x[t] /. soln)[[1]], {t, -2, 0}, PlotRange → {0, 3},
 PlotStyle → Thick, PlotLabel → "t-x Plane", AxesLabel → {"t", "x(t)"}]
```



```
Plot[(y[t] /. soln)[[1]], {t, -2, 0}, PlotRange → {-0.25`, 1},
 PlotStyle → Thick, PlotLabel → "t-y Plane", AxesLabel → {"t", "y(t)"}]
```



1. Repeat the steps above but replace the first equation by $\frac{dx}{dt} = 3x(t)^2 - y(t)$. Describe what happens.

It should be clear that once we get away from linear systems such as we considered above, *Mathematica* (or we) may not be able to solve the system analytically. When modeling physical phenomena by differential equations or systems, most of these systems or equations cannot be solved analytically. Thus we resort to numerical techniques. Let's start with the simplest: Euler's Method. We have already encountered this for first-order equations but now we must modify it to handle systems as well.

```
EulersMethod[t0_, x0_, y0_, h_, halt_] :=
  Module[{t1 = t0, x1 = x0, y1 = y0, j, m = halt},
   T1 = Table[t1 + j * h, {j, 0, m}];
   X1 = Table[x1, {j, 0, m}];
   Y1 = Table[y1, {j, 0, m}];
   For[j = 1, j ≤ m, j++,
    X1[[j + 1]] = X1[[j]] + h * f[T1[[j]], X1[[j]], Y1[[j]]];
    Y1[[j + 1]] = Y1[[j]] + h * g[T1[[j]], X1[[j]], Y1[[j]]];
   Return[Transpose[{T1, X1, Y1}]]];
```

In order to use the **EulersMethod** procedure on a system of the form

$$\frac{dx}{dt} = f(t, x, y)$$
$$\frac{dy}{dt} = g(t, x, y)$$

we need to input two functions and some initial conditions.

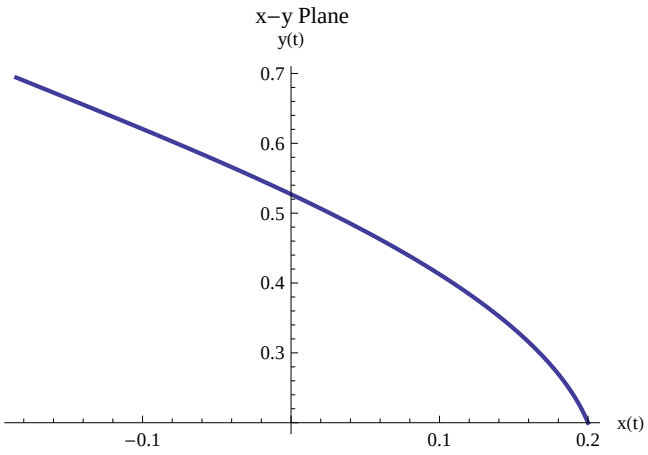
```
f[t_, x_, y_] = 3 * x^2 - y
g[t_, x_, y_] = x + y
```

$3x^2 - y$

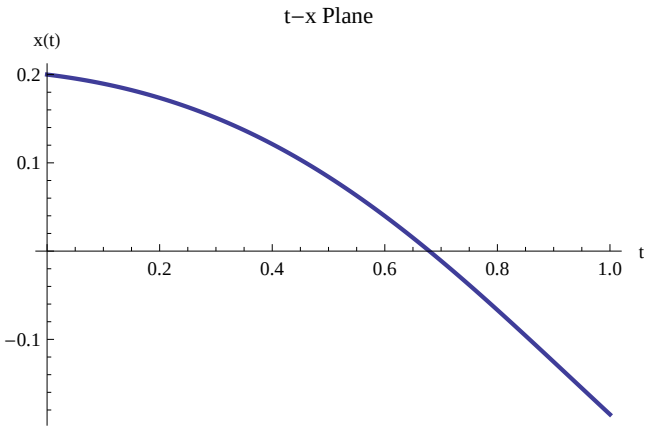
$x + y$

```
ptlist = EulersMethod[0, .2, .2, .01, 100];
```

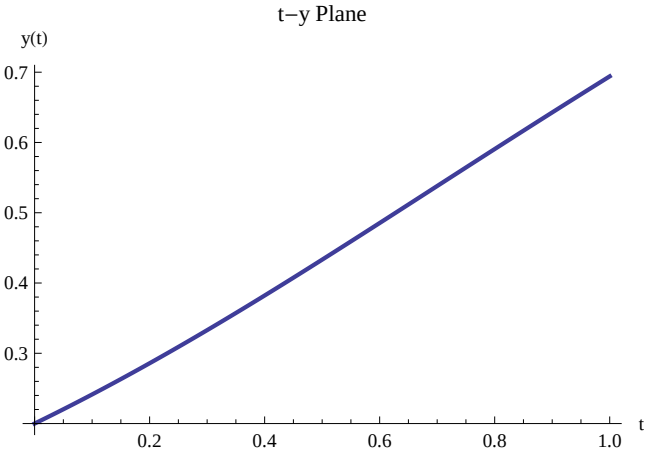
```
ptxy = Transpose[{ptlist[[All, 2]], ptlist[[All, 3]]}];
ListPlot[ptxy, Joined → True, PlotStyle → Thick,
 PlotLabel → "x-y Plane", AxesLabel → {"x(t)", "y(t)"}]
```




```
pttx = Transpose[{ptlist[[All, 1]], ptlist[[All, 2]]}];
ListPlot[pttx, Joined → True, PlotStyle → Thick,
 PlotLabel → "t-x Plane", AxesLabel → {"t", "x(t)"}]
```




```
ptty = Transpose[{ptlist[[All, 1]], ptlist[[All, 3]]}];
ListPlot[ptty, Joined → True, PlotStyle → Thick,
 PlotLabel → "t-y Plane", AxesLabel → {"t", "y(t)"}]
```



13.2 A Variation of Euler's Method

One slight variation of Euler's method is to replace the approximation of the derivative by a closer approximation. Recall that Euler's method relies on the fact that if (t_n, x_n) is a point on the graph of an unknown function *x(t)*, then the derivative $\frac{dx}{dt} = f(t, x)$ at the point (t_n, x_n) can be used to approximate the coordinate (t_{n+1}, x_{n+1}). Setting the derivative equal to its approximate value, that is,

$$\frac{x(t_n + h) - x(t_n)}{t_{n+1} - t_n} = \frac{x_{n+1} - x_n}{t_{n+1} - t_n} = \frac{dx}{dt} = f(t_n, x_n)$$

provided t_{n+1} is close to t_n, and clearing the fractions yields $x_{n+1} - x_n = (t_{n+1} - t_n)\frac{dx}{dt} = f(t_n, x_n)$, or equivalently, $x_{n+1} = x_n + h\frac{dx}{dt} = x_n + h f(t_n, x_n)$where $h = t_{n+1} - t_n$. Now instead of using the first-order difference equation

$$\frac{x(t_n + h) - x(t_n)}{t_{n+1} - t_n} = \frac{dx}{dt}$$

for the approximation of the derivative, we can use the second-order difference equation

$$\frac{x(t_n + h) - x(t_n - h)}{t_{n+1} - t_{n-1}} = \frac{dx}{dt}$$

for the approximation. This then means $x_{n+1} = x_{n-1} + 2h\frac{dx}{dt} = x_{n-1} + 2h f(t_n, x_n)$where $h = t_{n+1} - t_n$. This means to calculate x_{n+1} we must use two previous values.

Here is the new procedure with this modification. Notice that we are using new variable names since we wish to retain the values t1, x1, and y1 from the Euler's method that were previously obtained. Notice that we use the first order scheme to get the first value of x and y.

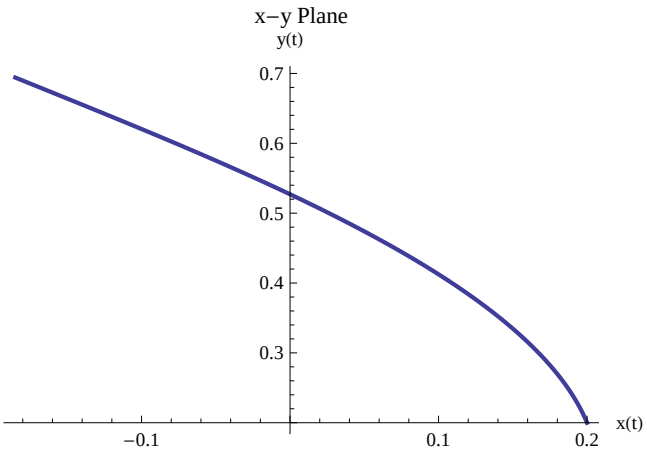
```
RevisedEulersMethod[t0_, x0_, y0_, h_, halt_] :=
  Module[{t2 = t0, x2 = x0, y2 = y0, j, m = halt},
   T2 = Table[t2 + j * h, {j, 0, m}];
   X2 = Table[x2, {j, 0, m}];
   Y2 = Table[y2, {j, 0, m}];
   X2[[2]] = X2[[1]] + h * f[T2[[1]], X2[[1]], Y2[[1]]];
   Y2[[2]] = Y2[[1]] + h * g[T2[[1]], X2[[1]], Y2[[1]]];
   For[j = 2, j ≤ m, j++,
    X2[[j + 1]] = X2[[j - 1]] + 2 * h * f[T2[[j]], X2[[j]], Y2[[j]]];
    Y2[[j + 1]] = Y2[[j - 1]] + 2 * h * g[T2[[j]], X2[[j]], Y2[[j]]];
   Return[Transpose[{T2, X2, Y2}]];

f[t_, x_, y_] = 3 * x^2 - y
g[t_, x_, y_] = x + y
```

$3x^2 - y$

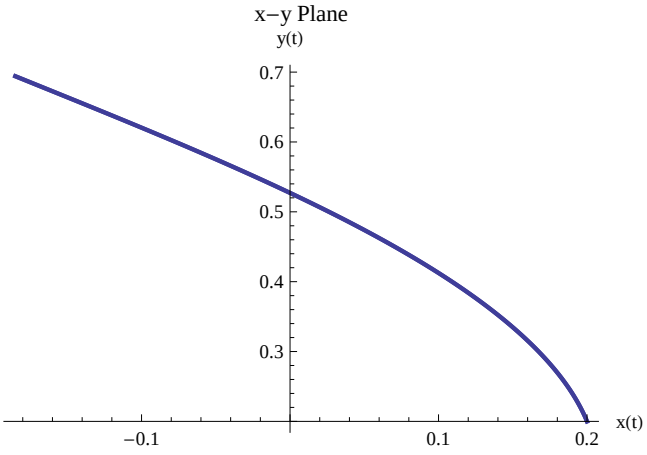
$x + y$

```
revptlist = RevisedEulersMethod[0, 0.2, 0.2, 0.01, 100];
revptxy = Transpose[{ptlist[[All, 2]], ptlist[[All, 3]]}];
ListPlot[revptxy, Joined → True, PlotStyle → Thick,
 PlotLabel → "x-y Plane", AxesLabel → {"x(t)", "y(t)"}]
```



2. Plot on the same *x-y* plane the curves for the Euler's method and the revised Euler's method and compare. Is there an appreciable difference? You may also wish to compare the output of the two procedures directly.

```
g1 = ListPlot[ptxy, Joined → True, PlotStyle → Thick,
    PlotLabel → "x-y Plane", AxesLabel → {"x(t)", "y(t)"}];
g2 = ListPlot[revptxy, Joined → True, PlotStyle → Thick,
    PlotLabel → "x-y Plane", AxesLabel → {"x(t)", "y(t)"}];
Show[
 g1,
 g2]
```



13.3 Improved Euler's Method

We can improve upon Euler's method by replacing the slope by the average of two slopes. The equations become,

$$\begin{aligned} t_{n+1} &= t_n + h \\ x_{n+1} &= x_n + \frac{h}{2}\left(f(t_n, x_n, y_n) + f(t_{n+1}, x_{n+1}, y_{n+1})\right) \end{aligned}$$, which by Euler's approximation becomes

$$x_{n+1} = x_n + \frac{h}{2}\left(f(t_n, x_n, y_n) + f(t_n + h, x_n + h\,f(t_n, x_n, y_n), y_n + h\,f(t_n, x_n, y_n))\right).$$

A similar equation holds for y_{n+1}.

Notice that we can compute successive values of *t* , *x*, and *y* from preceding values. We can incorporate this into a procedure similar to the procedures given above.

3. Complete the procedure for the improved Euler's method by filling in the correct equations for **Y3[[j+1]]**.

```
ImprovedEulersMethod[t0_, x0_, y0_, h_, halt_] :=
  Module[{t3 = t0, x3 = x0, y3 = y0, j, m = halt},
   T3 = Table[t3 + j * h, {j, 0, m}];
    X3 = Table[x3, {j, 0, m}];
    Y3 = Table[y3, {j, 0, m}];
    For[j = 1, j ≤ m, j++,
     X3[[j + 1]] = X3[[j]] + (h / 2) * (f[T3[[j]], X3[[j]], Y3[[j]]] + f[T3[[j]] + h, X3[[j]] +
            h * f[T3[[j]], X3[[j]], Y3[[j]]], Y3[[j]] + h * f[T3[[j]], X3[[j]], Y3[[j]]]);
     Y3[[j + 1]] = fill;
     Return[Transpose[{T3, X3, Y3}]]];

f[t_, x_, y_] = 3 * x^2 - y
g[t_, x_, y_] = x + y
```

$$3x^2 - y$$

$$x + y$$

```
impptlist = RevisedEulersMethod[0, 0.2, 0.2, 0.01, 100];
impptxy = Transpose[{ptlist⟦All, 2⟧, ptlist⟦All, 3⟧}];
g3 = ListPlot[impptxy, Joined → True, PlotStyle → Thick,
  PlotLabel → "x-y Plane", AxesLabel → {"x(t)", "y(t)"}]
```

4. How do the results for the improved Euler's method compare with the revised Euler's method? What is the difference in the values of x and y for the two methods at t = 1? To determine the answers, complete the following commands as indicated and interpret your results.

```
Show[g2, g3]
revptlist⟦101⟧
impptlist⟦101⟧
```

13.4 The Runge-Kutta Method

Among the various methods, there are built-in numeric methods in *Mathematica*. One is the fourth-order Runge-Kutta method. We can employ it by calling on the **NDsolve** command and specifying the method as the Runge-Kutta fourth-order method. (See Section 5.3 of the text by Nagle, Saff, and Snider for the algorithm of the Runge-Kutta method.)

```
f[t_, x_, y_] = 3 * x^2 - y;
g[t_, x_, y_] = x + y;
diffeq1 = D[x[t], t] == f[t, x[t], y[t]]
diffeq2 = D[y[t], t] == g[t, x[t], y[t]]
```

$$x'[t] == 3\,x[t]^2 - y[t]$$

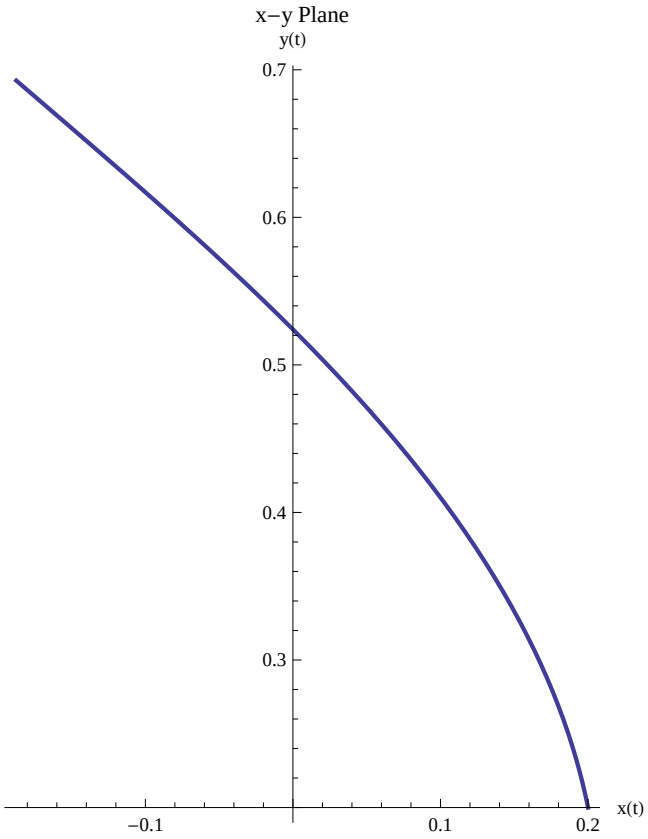
$$y'[t] == x[t] + y[t]$$

```
soln = NDSolve[{diffeq1, diffeq2, x[0] == .2, y[0] == .2}, {x[t], y[t]},
   {t, 0, 1}, Method → {"ExplicitRungeKutta", "DifferenceOrder" → 4}];

xsoln = x[t] /. soln;
ysoln = y[t] /. soln;
Table[{tval, xsoln /. t → tval, ysoln /. t → tval}, {tval, 0, 1, .1}] // TableForm
```

0.	0.2	0.2
0.1	0.189374	0.241566
0.2	0.172922	0.286093
0.3	0.149859	0.333236
0.4	0.119605	0.382544
0.5	0.0819342	0.433472
0.6	0.0371294	0.48542
0.7	−0.013921	0.537787
0.8	−0.0697202	0.590031
0.9	−0.128296	0.641738
1.	−0.187454	0.692674

```
rkplot = ParametricPlot[{soln[[1, 1, 2]], soln[[1, 2, 2]]}, {t, 0, 1},
  PlotStyle → Thick, PlotLabel → "x-y Plane", AxesLabel → {"x(t)", "y(t)"}]
```



13.5 Exercises

5. We now have employed four different numerical methods to solve the same two-dimensional system. Consider now the following system.

$$\frac{dx}{dt} = x + y + e^{-t}$$
$$\frac{dy}{dt} = x - y$$

Use *Mathematica* to solve this system with initial value x(0) = 1, y(0) = -1 for an analytic solution, and four numeric solutions. Compare each of the solutions at the value of t = 1 and see which is closest to the actual value. Make sure your step sizes are compatible.

6. Repeat Exercise #5 for the systems

$$\frac{dx}{dt} = x + y + t\sin t$$
$$\frac{dy}{dt} = x - y$$

and

$$\frac{dx}{dt} = x + y + t\sin t$$
$$\frac{dy}{dt} = x - y + \cos t$$

You may pick any reasonable initial conditions.

7. What would you conclude is the most accurate numerical method from your observations?

8. Consider now the equation for the harmonic oscillator

$$m\frac{d^2y}{dt^2} + k_d\frac{dy}{dt} + k_s y = 0$$

Convert this into a system of first-order equations. Use what you believe to be the most reliable numerical technique above to complete the following.

(a) First take $k_d = 0$, $m = 2$ and 3 and $k_s = 5$ or 6 and obtain the graphs of the solutions in the *x-y* plane, the *t-x* plane, and the *t-y* plane. Describe the behavior of the solutions and give the physical interpetation in each case.

(b) Now include damping by letting $k_d = 3$ and repeat part (a). Describe the behavior of the solutions and give the physical interpretation in each case.

9. Using the Runge-Kutta numeric method, approximate the solution to the initial-value problem

$$3t^2y'' - 5ty' + 5y = 0,\ y(1) = 0,\ y'(1) = \frac{2}{3}$$

at the value of t = 8. Use stepsizes of .1 and .01 and compare your answers to the actual solution $y = t^{5/3} - t$.

Laboratory Fourteen

Series Solutions

(Sections 8.1 - 8.4 of the Nagle/Saff/Snider text)

In previous labs, we have examined various analytical methods and numeric methods to solve linear differential equations. Linear, homogeneous equations with constant coefficients are particularly easy to solve by examining the roots of the auxiliary equation. However, linear equations whose coefficients are functions instead of constants are instances where other techniques are needed. One appropriate technique to use is to find a series solution.

In this section we will generate series solutions to initial-value problems of the form

$$\frac{d^2 x(t)}{d t^2} + p(t)\frac{d x}{d t} + q(t)\,x(t) = 0,\ x(t_0) = 0,\ \text{and}\ x'(t_0) = x_1$$

Recall that if $p(t)$ and $q(t)$ are both analytic at t_0 , then t_0 is called an *ordinary* point for the above differential equation. At an ordinary point, we can always generate a general solution of the above equation by means of series. This is contained in the following existence theorem.

Theorem 1. Suppose t_0 is an ordinary point for the above equation. Then the above equation has two linearly independent analytic solutions of the form

$$x(t) = \sum_{n=0}^{\infty} a_n (t - t_0)^n$$

Moreover, the radius of convergence of any power series solution of the form given by this equation is at least as large as the smaller of the two radii of convergence for the series for $p(t)$ and $q(t)$.

Our objectives in this section are:

1. To learn how to construct power series solutions using *Mathematica.*

2. To see how these series converge to the solution and compare with closed form solutions.

3. To see how two linearly independent series solutions can be generated by *Mathematica*.

14.1 Examples

Let's begin with a simple example.

Example 1. Consider the differential equation $\frac{d\,x(t)}{d\,t} + t\,x(t) = 0$. Find a series solution to this first-order equation and for the initial-value problem with $x(0) = 1$ involving this equation. Determine to what function the series converges. How does the series solution compare with the closed form of the solution?

```
eqn = x'[t] + t x[t] == 0;
lhs = Series[eqn[[1]], {t, 0, 5}]
coeffs = Solve[{lhs == 0, x[0] == 1}]
Series[x[t], {t, 0, 5}] /. coeffs
```

$$x'[0] + (x[0] + x''[0])\,t + \left(x'[0] + \frac{1}{2}x^{(3)}[0]\right)t^2 + \frac{1}{6}\left(3x''[0] + x^{(4)}[0]\right)t^3 + \frac{1}{24}\left(4x^{(3)}[0] + x^{(5)}[0]\right)t^4 + \frac{1}{120}\left(5x^{(4)}[0] + x^{(6)}[0]\right)t^5 + O[t]^6$$

$$\left\{\left\{x[0] \to 1,\ x'[0] \to 0,\ x''[0] \to -1,\ x^{(3)}[0] \to 0,\ x^{(4)}[0] \to 3,\ x^{(5)}[0] \to 0,\ x^{(6)}[0] \to -15\right\}\right\}$$

$$\left\{1 - \frac{t^2}{2} + \frac{t^4}{8} + O[t]^6\right\}$$

Notice that this is the beginning of a power series about the point $t = 0$. To get more terms of the series we can increase the order.

```
eqn = x'[t] + t x[t] == 0;
lhs = Series[eqn[[1]], {t, 0, 11}]
coeffs = Solve[{lhs == 0, x[0] == 1}]
ssoln = Series[x[t], {t, 0, 11}] /. coeffs
```

$$x'[0] + (x[0] + x''[0])\,t + \left(x'[0] + \frac{1}{2}x^{(3)}[0]\right)t^2 + \frac{1}{6}\left(3x''[0] + x^{(4)}[0]\right)t^3 + \frac{1}{24}\left(4x^{(3)}[0] + x^{(5)}[0]\right)t^4 + \frac{1}{120}\left(5x^{(4)}[0] + x^{(6)}[0]\right)t^5 + \frac{1}{720}\left(6x^{(5)}[0] + x^{(7)}[0]\right)t^6 + \frac{\left(7x^{(6)}[0] + x^{(8)}[0]\right)t^7}{5040} + \frac{\left(8x^{(7)}[0] + x^{(9)}[0]\right)t^8}{40\,320} + \frac{\left(9x^{(8)}[0] + x^{(10)}[0]\right)t^9}{362\,880} + \left(\frac{x^{(9)}[0]}{362\,880} + \frac{x^{(11)}[0]}{3\,628\,800}\right)t^{10} + \left(\frac{x^{(10)}[0]}{3\,628\,800} + \frac{x^{(12)}[0]}{39\,916\,800}\right)t^{11} + O[t]^{12}$$

$$\left\{\left\{x[0] \to 1,\ x'[0] \to 0,\ x''[0] \to -1,\ x^{(3)}[0] \to 0,\ x^{(4)}[0] \to 3,\ x^{(5)}[0] \to 0,\ x^{(6)}[0] \to -15,\ x^{(7)}[0] \to 0,\ x^{(8)}[0] \to 105,\ x^{(9)}[0] \to 0,\ x^{(10)}[0] \to -945,\ x^{(11)}[0] \to 0,\ x^{(12)}[0] \to 10\,395\right\}\right\}$$

$$\left\{1 - \frac{t^2}{2} + \frac{t^4}{8} - \frac{t^6}{48} + \frac{t^8}{384} - \frac{t^{10}}{3840} + O[t]^{12}\right\}$$

In this case *Mathematica* may be able to get an analytic solution as well. Let's try.

```
soln = DSolve[{eqn, x[0] == 1}, x[t], t]
```

$$\left\{\left\{x[t] \to e^{-\frac{t^2}{2}}\right\}\right\}$$

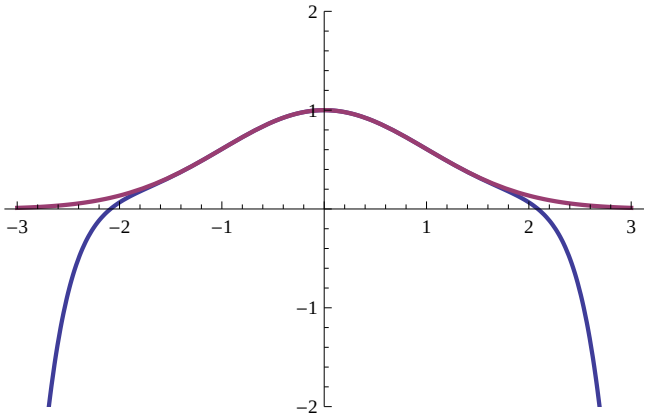
It is easy to check that the Taylor (Maclaurin) series for this function $x(t)$ is indeed the series given above and that we thus have our unique solution in either series form or closed form. (To check this you can also use the **Series** command. See help on **Series**.)

It is instructive to plot on the same graph the series solution and the closed form exact solution. To do this we must first convert the series solution into a polynomial, in this case a 10th degree polynomial.

```
polysoln = Normal[ssoln][[1]]
```

$$1 - \frac{t^2}{2} + \frac{t^4}{8} - \frac{t^6}{48} + \frac{t^8}{384} - \frac{t^{10}}{3840}$$

```
Plot[{polysoln, x[t] /. soln}, {t, -3, 3}, PlotRange → {-2, 2}, PlotStyle → Thick]
```



As we can see the graphs are almost identical at least on the interval $[-1, 1]$ but for $|t| > 2$, they really start to diverge.

To actually see how close the values are we can print a table of the values.

```
Chop[Table[{t, soln[[1, 1, 2]], polysoln}, {t, -2, 2, .2}]] // N // TableForm
```

```
-2.   0.135335  0.0666667
-1.8  0.197899  0.177609
-1.6  0.278037  0.27289
-1.4  0.375311  0.374234
-1.2  0.486752  0.486577
-1.   0.606531  0.60651
-0.8  0.726149  0.726148
-0.6  0.83527   0.83527
-0.4  0.923116  0.923116
-0.2  0.980199  0.980199
0.    1.        1.
0.2   0.980199  0.980199
0.4   0.923116  0.923116
0.6   0.83527   0.83527
0.8   0.726149  0.726148
1.    0.606531  0.60651
1.2   0.486752  0.486577
1.4   0.375311  0.374234
1.6   0.278037  0.27289
1.8   0.197899  0.177609
2.    0.135335  0.0666667
```

Example 2. Consider now the initial value problem $\frac{d^2x}{dt^2} + \frac{1}{t}\frac{dx}{dt} - \frac{x}{t^2+1} = 0,\ x(-1) = c_1,$ and $x'(-1) = c_2$. We begin by finding the series solution for $c_1 = 1, c_2 = 0$.

```
eqn = x''[t] + (1 / t) x'[t] - x[t] / (t^2 + 1) == 0;
lhs = Series[eqn[[1]], {t, -1, 5}];
coeffs = Solve[{lhs == 0, x[-1] == 1, x'[-1] == 0}];
ssoln1 = Series[x[t], {t, -1, 5}] /. coeffs
```

$$\left\{1+\frac{1}{4}(t+1)^2+\frac{1}{6}(t+1)^3+\frac{11}{96}(t+1)^4+\frac{1}{12}(t+1)^5+O[t+1]^6\right\}$$

To actually see the two linearly independent solutions we could now let $c_1 = 0$,and $c_2 = 1$ and solve again:

```
eqn = x''[t] + (1 / t) x'[t] - x[t] / (t^2 + 1) == 0;
lhs = Series[eqn[[1]], {t, -1, 5}];
coeffs = Solve[{lhs == 0, x[-1] == 0, x'[-1] == 1}];
ssoln2 = Series[x[t], {t, -1, 5}] /. coeffs
```

$$\left\{(t+1)+\frac{1}{2}(t+1)^2+\frac{5}{12}(t+1)^3+\frac{1}{3}(t+1)^4+\frac{127}{480}(t+1)^5+O[t+1]^6\right\}$$

The beginning terms of two linearly independent solutions are evident. The solution to the initial value problem may thus be written as

$$c_1\left(1+\frac{1}{4}(t+1)^2+\frac{1}{6}(t+1)^3+\frac{11}{96}(t+1)^4+\frac{1}{12}(t+1)^5+O(t+1)^6\right)+$$
$$c_2\left((t+1)+\frac{1}{2}(t+1)^2+\frac{5}{12}(t+1)^3+\frac{1}{3}(t+1)^4+\frac{127}{480}(t+1)^5+O(t+1)^6\right)$$

14.2 Exercises

1. According to Theorem 1, what should be the radius of convergence of a power series solution to the initial value problem of Example 1? Demonstrate that the interval of convergence contains the interval $[-6, 6]$ by choosing the order of the series (or the polynomial) large enough so that the graph and table of values for **soln** and **polysoln** essentially agree on the interval $[-6, 6]$. (Be careful! You may wish to change the increment for the table so that you do not get too many data points and also suppress the actual long expression for **polysoln**.) Do you think you could do the same for on any interval $[-c, c]$?

2. According to Theorem 1, what is at least the radius of convergence (or the interval of convergence) of a power series solution to the initial-value problem of Example 2? Pick values for c_1 and c_2 and generate the graphs of the solution for various orders. Do the graphs and the tables of values indicate convergence on the interval that you have chosen?

3. Consider the initial-value problem $\frac{dx}{dt} - 2x = \sin x,\ x(0) = 1$.

(a) See if *Mathematica* can find a closed form solution.

```
eqn = x'[t] - 2 x[t] == Sin[x[t]];
DSolve[{eqn, x[0] == 1}, x[t], t]
```

(b) Let *Mathematica* find series solutions of order 4, 6, and 12. Fill in the blanks marked **fill** with appropriate data.

```
eqn = x'[t] - 2 x[t] == Sin[x[t]];
lhs = Series[eqn[[1]], {t, 0, fill}]
coeffs = Solve[{lhs == 0, x[0] == 1}]
ssoln = Series[x[t], {t, 0, fill}] /. coeffs
```

(c) Graph each of these solutions on the same graph for the interval [-1,1].

(d) Chart the values of the solutions on the same chart for t in the interval [-1, 1] and increment .2.

4. Find series solutions of orders 6 and 12 for each of the following initial-value problems. Convert each solution into a polynomial so that you can graph the solutions for orders 6 and 12 on the same coordinate axis.

(a) $\frac{d^2x}{dt^2} + t\,\frac{dx}{dt} - t^2 x = \ln(t+1),\ x(0) = 2,\ \text{and}\ x'(0) = 1$

(b) $\left(t^2 - 1\right)\frac{d^2x}{dt^2} + 2\,\frac{dx}{dt} - 4\,t\,x = 0,\ x(0) = 5,\ \text{and}\ x'(0) = -2$

5. **Legendre's Equation** (Section 4.7 of text by Nagle/Saff/Snider). The equation $\left(1 - x^2\right)\frac{d^2y}{dx^2} - 2\,x\,\frac{dy}{dx} + n(n+1)\,y = 0$ where *n* is an unspecified parameter is called Legendre's equation.

(a) Find a power series expansion about x = 0 for a solution of Legendre's equation with initial values $y(0) = 1$and $y'(0) = 0$.

(b) For *n* a nonnegative integer there exists an *n th* degree polynomial that is a solution to Legendre's equation. These are called Legendre polynomials. For *n* equal to 5 and 10 show that this is true and determine the Legendre polynomials for both 5 and 10. (Hint: Choose initial values y(0) = *a* and y'(0) = *b*, pick the order large enough, and generate the series solution.)

6. **Aging Spring** (Section 8.4 of text by Nagle/Saff/Snider). As a spring ages, its spring "constant" decreases in value. One such model for a spring-mass system with an aging spring is $m\,\frac{d^2x}{dt^2} + b\,\frac{dx}{dt} + k\,e^{-\eta t}\,x = 0$, where *m* is the mass, *b* is the damping constant, *k* and η are positive constants, and $x(t)$ is the displacement of the spring from its equilibrium position.

(a) Let *m* = 1 kg, *b* = 2 N-sec/m, *k* = 1 N/m, and η = 1/sec. Set the system in motion by displacing the mass one meter from its equilibrium position and then releasing it ($x(0) = 1, x'(0) = 0$). Find 8 terms of a power series solution for the mass-spring system.

(b) Now take *b* = 0 and let η be an arbitrary positive number. Generate 8 terms of the power series solution with η as a parameter.

(c) What is the function represented by the power series that you obtained in part (b) if η is zero?

5. Additional Project Descriptions

5.1 Notes to the Instructor: Group Projects

One of the strengths of the Nagle/Saff/Snider texts is the number of projects that are included at the end of each chapter. The selection of projects is expanded in the new edition (Eighth and Sixth Editions, 2011). *Mathematica* can be used to assist with the analysis of many of these projects (e.g., **Laboratory Four - Picard's Method** contains a complete project report for Project B, Chapter 1.)

The seven additional projects presented in this supplement illustrate that a project can be designed around almost any topic. The projects in this supplement represent a diverse set of applications, including chemical engineering, mathematical epidemiology, special functions, electrical circuits, and athletics. Likewise, the analytical tools required to complete the investigation of these projects are varied. One of the models emphasizes the modeling process more than the solution of IVPs, and at least three of these projects involve systems of IVPs.

A brief overview of each project contained in this section is provided below. The actual descriptions comprise the remainder of this chapter. (There is no specific ordering to the projects.)

- **Design Your Own Project**

The basic idea here is for each student to find an application of differential equations related to a topic of personal interest - their major, another course in which they are currently enrolled, or something that has always fascinated them - and then prepare a project description, complete with a derivation of the model, questions to be addressed, and realistic data.

Then, after the project description is approved, the students have to prepare a project report for their project. Or, if working in groups, each group selects one group member's project for complete analysis.

- **The ODE of World-Class Sprints**

This project is ideal for use with Chapter 2. The differential equation is linear (and separable). The real interest in this project is the mathematical modeling. The application is easy to understand and discuss. The questions are not technically sophisticated, but some of the underlying assumptions and interpretations are not completely trivial.

Familiarity with least squares techniques is helpful for Step III of the project, but it is not necessary.

- **Consecutive Reactions for Batch Reactors**

This is another project that can be given very early in the semester. Even though the mathematical model is a system of linear ODEs, the system is "lower triangular" and can be solved by "forward substitution."

The specific constants used in this lab do not pertain to any real situations. It's not unreasonable to ask the students to research this problem and identify specific instances in which this model arises and to identify realistic parameter values.

This project originated with Ralph White, Chemical Engineering and Center for Electrochemical Engineering, University of South Carolina, as part of project funded by the Lilly Foundation and the USC Office of the Provost.

- **Normal T-Cells**

This project is Part 1 of a two-part problem. The second part is “T-Cells in the Presence of HIV” and is appropriate only after systems of ODEs have been discussed. This project requires only the basics of population modeling (constant, exponential, and logistic rates of change).

This project is an adaptation of the discussion presented by Herod and Yeagers (*MapleTech*, 1994), which is based on a model proposed by Perelson et al. (*Math. Biosc*., 1993).

- **T-Cells in the Presence of HIV**

This project is Part 2 of a two-part problem. The first part, “Normal T-Cells”, can be assigned much earlier than this project (which involves a system of ODEs).

A third project in this series can be constructed around the question of possible treatments for HIV/AIDS. An accessible discussion of this topic can be found in the paper “Using mathematics to understand HIV immune dynamics,” by Denise Kirschner (*Notices of the AMS*, v. 43, no. 2, February 1996, pp. 191--202).

- **Design of an Electrical Network**

This problem is based on a standard problem in electrical circuits. The focus on this project is a parametric analysis of the solution as a function of one of the resistances in the network. Both the steady-state solution and the manner in which the solution co verges to the steady-state solution depend on the value of the resistance.

This project requires knowledge about systems of linear ODEs. While it is ideally suited for solution using matrix methods (Chapter 9), the project can be modified (see the next comment) so that it is not unreasonable to solve without the use of linear algebra.

The application does require Kirchoff's voltage law, which is introduced in Section 3.5. Also, there are two values of the parameter for which the system does *not* have a full set of eigenvectors. The treatment of this case is handled in the Exercises for Section 9.5 and also in Section 9.8.

- **Hypergeometric Functions**

A nice introduction to hypergeometric functions is presented in Section 8.8. This introduction only begins to discuss the multitude of identities involving hypergeometric functions. This project continues this discussion and emphasizes more properties of these functions.

5.2 Design Your Own Project

- **Step I -- Identify the Application**

A variety of applications of differential equations are discussed in the text and in this laboratory manual. Included among the many applications are applications in Newtonian mechanics, population dynamics, electrical circuits, and mixing problems. The diversity of these applications illustrates the broad applicability of differential equations. To add emphasis to this point, this project requires that you identify an application of differential equations in a subject area of particular interest to you.

You are specifically encouraged to discuss this project with a project mentor. This mentor should be a professional scientist, e.g., a professor for another course you are presently taking, a course you will be taking next semester, or your academic advisor. Explain this assignment to the mentor, and provide a copy of this project assignment.

The application should be solvable by techniques discussed in this course. The only other requirement is that the model have at least one parameter. The equations should be ordinary differential equations but can be either initial-value problems or boundary-value problems. The equations can be of any (positive) order. The problem might involve a single equation or a system of equations.

- **Step II -- Background Investigation**

The main result of this assignment will be a brief description of your project in the form of a project assignment. Prior to writing this assignment, it will be necessary for you to thoroughly understand the problem and the main questions that are to be addressed. In consultation with your project mentor, you should identify a short list of references for your application. These references should be accessible to students in your class, but they do not need to be mathematical.

Study these references and pay special attention to the mathematical modeling used to convert the physical problem into a mathematical problem, any assumptions that are made during the modeling process, and the questions that will be the focus of the project description you will create. (Be sure to request assistance and clarifications from your mentor or differential equations instructor, as needed.)

- **Step III -- Project Description**

Prepare a project description for your project. This description, which should be no more than two pages, (one page is more than enough in most cases), must be typed (word processed) and may include illustrations or other graphics as needed.

The project description should contain all of the following: title, your name, name of project mentor, brief description of the physical problem, summary of the mathematical modeling, questions to be answered in the analysis, realistic values for all parameters, and a list of at least two references related to the project.

- **Step IV -- Summary and Conclusion**

This project is longer than it sounds - unless you start early and divide the assignment into manageable portions. Even though you may not have to answer the questions you pose for this project, your next step will be to answer the questions in one of the projects proposed by one of your group members. Thus, it is important that you select relevant, interesting, and tractable questions. This may take a few iterations before you find the appropriate level. You may need to discuss your ideas with your instructor and ask for help when needed.

Each group member should describe his or her project to the group. The group then needs to select one of the projects for further analysis. Be sure that all questions are answered and that final report is similar in style to the other project reports prepared for this course.

5.3 The ODE of World-Class Sprints

■ Step I -- Understanding the Model

According to Keller's theory of competitive running, published in 1973, track sprints of up to 300 meters can be described by the following differential equation.

$$\frac{dv}{dt} = A - \frac{v}{\tau} \quad (1)$$

At the time of this analysis, the best constants were A=12.2 m/sec and τ = 0.892 sec.

● What is an appropriate initial condition for the problem?

● In your own words, explain each of the terms in the differential equation and how the terms combine to form a complete model. What is an interpretation of the two parameters A and τ ? Does the differential equation seem reasonable in your experience? Do you know any other situations that could be described using this same model?

■ Step II -- Finding and Interpreting Solutions

One reason for solving differential equations is to derive further consequences from the solution. The next set of questions explores the solution and its consequences.

● Solve the initial value problem for $v(t)$ symbolically in terms of the parameters A and τ.

● Find the distance $D(t)$ traveled in a sprint as a function of time symbolically in terms of A and τ.

● Find the acceleration function $a(t)$ as a function of time. Also, what differential equation does $a(t)$ satisfy? What are the initial conditions?

● Draw a graph of the distance, velocity, and acceleration over a reasonable time interval using the 1973 parameters.

● When does the maximum acceleration occur for the 1973 parameters? How long does it take for the acceleration to drop to 10% of its maximum value?

● What is the runner's maximum speed? How long does it take for the runner to reach 90% of this maximum. Compare the final speeds in a 100m and a 200m race. Explain your findings.

● Answer the previous two questions for general values of A and τ.

■ Step III -- Extending the Problem (Data Fitting)

Another aspect of differential equations is to use the solution to develop the parameters of the problem. This is a mathematical form of the game *Jeopardy*. Given the solution (or the data), find the equation (or the parameters for the equation). In more advanced applications of differential equations, mathematicians and scientists call this an *inverse problem*. Inverse problems can be very difficult and are an active area of research in applied mathematics.

Race officials often record the split times of runners in addition to the final times. Split times are the measure of time required to cover portions of the distance. In Table 1 are the results for the top four men and women in the 100m sprint at the 1993 World Track and Field Championships held in Stuttgart, Germany.

● To find out if athletic performance has changed since 1973, determine values for the parameters A and τ using the 1993 results.

● Compare the A and τ values of the men and women. Some track and field experts speculate that men's and women's abilities are becoming identical. Do your conclusions support this?

■ Step IV -- Summary and Conclusion

The theoretical analysis and the work that you have done with the data should give you a better insight into this model. To conclude the project, let's consider a few of the following points.

● Is it preferable to accelerate quickly at the start and then maintain the speed or to hold back and accelerate over a longer period of time? How would the model change to allow for weakening, i.e. deceleration, towards the end of the race? (This answer should be in the form of instructions, and explanation, to the sprinter and/or their coach.)

● Are the results produced by the model reasonable? Why does Keller suggest that this model is valid only for races up to 300m? Do you agree with this? What types of modifications would be needed for longer races (e.g., 1600m, 10km, or marathon)?

Name	Nation	30 meters	60 meters	80 meters	100 meters
Linford Christie	GBR	3.85	6.45	8.15	9.87
Andre Cason	USA	3.83	6.43	8.15	9.92
Dennis Mitchell	USA	3.82	6.46	8.22	9.99
Carl Lewis	USA	3.95	6.59	8.30	10.02
Gail Devers	USA	4.09	6.95	8.86	10.82
Merlene Ottey	GHA	4.13	6.98	8.87	10.82
Gwen Torrence	USA	4.14	7.00	8.92	10.89
Irina Privalova	RUS	4.09	7.00	8.96	10.96

Table 1. Split times for 100m sprint, 1993 World Track and Field Championships.
Data courtesy of Shawn Price of *Track and Field News*.

■ Bibliography

[1] Alexandrov, I. and Lucht, P., Physics of Sprinting, *American Jour. Physics*, 49, 1977.

[2] Dunbar, Steven R., The ODE of World-Class Sprints, *C-ODE-E Newsletter*, Spring 1994, p.7-9.

[3] Keller, J.B., A Theory of Competitive Running, *Physics Today*, Sept. 1973, p. 43.

[4] Pritchard, W.G., Mathematical Models of Running, *SIAM Review*, Sept. 1992, 35, pp.359-379.

[5] Thompson, W.J., *Computing in Applied Science*, Wiley, 1984, p. 107-109.

5.4 Consecutive Reactions for Batch Reactors

- **Step I -- Understanding the Problem**

Consider a reaction involving three substances, A, R , and S , which have first-order reaction rates and are carried out isothermally in either the gas or liquid phase in a perfectly mixed batch reactor. The reaction begins with a pure sample of component A. Schematically, the process can be represented as:

$$A \to R \qquad r_A = k_1 C_A$$
$$R \to S \qquad r_S = k_2 C_S$$

where r_A is the reaction rate for the conversion of A into R , r_S is the rate at which R is converted into S ; C_A and C_S are the *concentrations* of A and S , respectively.

The primary objective is to determine the amount of each substance in the reactor up to time after the reaction starts. Additional information that is desired includes the time at which the amount of substance R is a maximum, the time at which the amount of S exceeds the amount of A and the total amount of substance S produced at any time $t > 0$.

- **Step II -- Mathematical Formulation/Modeling**

This is really just a three-compartment container problem. Looking at the inflow and outflow for each substance leads to a system of three first-order differential equations with initial conditions. Let N_A,N_R, and N_S denote the number of moles of A , R , and S at any time in the reactor. The volume of solution in the tank is assumed to be constant. Then, conservation of the total amount of each of the three components can be expressed as

$$\frac{dN_A}{dt} = -k_1 N_A$$
$$\frac{dN_R}{dt} = k_1 N_A - k_2 N_R$$
$$\frac{dN_S}{dt} = k_2 N_R$$

with initial conditions $N_A(0) = 1$, $N_R(0) = 0$, and $N_S(0) = 0$. Identify all mathematical information that is needed to answer the questions posed at the end of Step I. (Assume only that you know explicit formulae for N_A,N_R, and N_S.)

- **Step III -- Solution and Analysis**

Use the ideas from first-order linear equations to find the solution, N_A,N_R, and N_S, to this system. Be sure to find solutions for all possible combinations of k_1 and k_2 (both positives). Use your solution, with $k_1 = 1$ and $k_2 = 3$, to answer the questions posed a the end of Step I. Answer the same questions with $k_1 = 3$ and $k_2 = 1$. Are there any differences between the cases when $k_1 < k_2$ and $k_1 > k_2$? Does anything special happen when $k_1 = k_2$?

- **Step IV -- Summary and Conclusion**

Use your findings from Step III to provide complete answers to the questions in Step I for any positive constants k_1 and k_2.

Bibliography

[1] Holland, C.D. and Anthony R.G., *Fundamentals of Chemical Reaction Engineering*, Prentice-Hall, 1979.

5.5 Normal T-Cells (Part 1 of 2)

- **Step I -- Background**

Perelson, Kerschner, and DeBoer [4] have presented a model for the interaction of T-cells and HIV. In this project we will examine a preliminary model that explains the normal functioning of T-cells. The model of T-cell formation in the presence of HIV is contained in the project titled “T-Cells in the Presence of HIV”.

This project has a slightly different structure than most of the others. The general idea is to explain a biological theory. Mathematical problems will be encountered, formulated, solved, and interpreted throughout this investigation. It might be even more informative if you compared this discussion with the discussion found in the original paper.

- **Step II -- The Basic Model**

T-cells are a natural and essential component of the human immune system. The primary job of T-cells is to attack and destroy cancerous cells, cells infected with a virus, and, in general, any cells containing foreign antigens. Prior to examining the interaction between T-cells and HIV, it is necessary to understand a little about how T-cells function under normal conditions.

The typical T-cell count in a healthy person is about 1000 to 1200 per cubic millimeter. T-cells are created in the lymphatic tissues (including the thymus). These tissues normally produce T-cells at a uniform rate. When the body detects a need for T-cells to combat an infection, additional T-cells are created. T-cells only live for a finite period of time.

A mathematical model for the number of T-cells in the body as a function of time (t) must account for the two different mechanisms controlling the production of T-cells and the death of T-cells as they age. Assuming a logistic growth model for the need-based production of T-cells, and a constant per-capita death rate, this model would take the form

$$f(T) = s + r\,T\left(1 - \frac{T}{T_{max}}\right) - \mu\,T$$

and the parameters s, r, $T_{\max}$, and μ are all positive.

- **Step III -- Understanding the Basic Model**

Identify each of the three terms in the RHS of this differential equation and explain the meaning of each of the parameters (What are units associated with each of the parameters? Assume time is measured in days.)

Explain why the conditions $f(0) > 0$ and $f(T_{\max}) < 0$ assure that the number of T-cells will stay between 0 and $T_{\max}$. Under what additional constraints on the parameters are these conditions satisfied?

The fact that f is continuous on $[0, T_{\max}]$ and changes sign on this interval implies, by the Intermediate Value Theorem, that there is some value of T, say T^*, for which $f(T^*) = 0$.

What is the significance of this number of T-cells? Find T^* in terms of the parameters s, r, $T_{\max}$, and μ.

- **Step IV -- Qualitative Analysis for a Specific Case**

Realistic value of the parameters for a healthy individual are $s = 10$, $r = 0.03$, $T_{max} = 1700$, and $\mu = 0.02$. Do these parameters satisfy the constraints identified earlier? What is T^*? Is this consistent with the biological descriptions?

Determine the qualitative behavior of solutions for all initial conditions between 0 and T_{max} for the specific values of the parameters given. Do all solutions stay in the desired interval?

- **Step V -- Qualitative Analysis for the General Case**

Repeat Step II for general values of the parameters.

- **Bibliography**

[1] Greene, W.C., AIDS and the Immune System, *Scientific American*, September 1993 (Special Issue).

[2] Herod, J.V., A Model for an HIV Infection, *Maple Technical Newsletter*, 10, 1993, p.72-78.

[3] Herod, J.V., Yeagers, E. K., Mathematics, Biology, and Maple, *Maple Technical Newsletter* (Special Issue), 1994, p. 69-76.

[4] Perelson, A.S., Kerschner D.E., DeBoer R., Dynamics of HIV Infections of $CD4_+$ T-Cells, *Mathematical Biosciences*, 114, 1993, p. 81-125.

5.6 T-Cells in the Presence of HIV (Part 2 of 2)

Step I -- Background

Perelson, Kerschner, and DeBoer [4] have presented a model for the interaction of T-cells and HIV. In the project "Normal T-Cells" we learned how T-cells function in a healthy individual. In this project we will see some of the changes that are needed to model T-cells when the HIV is present in the body. This discussion is more difficult than the previous one. Thinking about the biological interpretation of each term should help you understand the full model. The final qualitative results should be consistent with what you know about T-cells, HIV, and AIDS.

This project has a slightly different structure than most of the others. The general idea is to explain a biological theory. Mathematical problems will be encountered, formulated, solved, and interpreted throughout this investigation. It might be even more informative if you compared this discussion with the discussion found in the original paper.

Step II -- The Basic Model

The HIV virus attacks various types of T-cells. The T-cells are able to fight this invasion for a period of time, but eventually cannot withstand the onslaught. As a result, the immune system becomes less effective at fighting diseases (including the spread of HIV). The models presented in this section of the project will incorporate some of the forces involved in this process.

The first difference is that it is now necessary to include v , the amount of HIV, in our analysis. Also, there will be a distinction between T-cells that are not infected, T , T-cells that are latently infected, T_L , and the T-cells that are actively infected, T_A. (Latently infected T-cells are those that are infected with the provirus and are not producing free virus; actively infected T-cells are infected and are producing free virus.)

A different differential equation will be needed for each of v , T , T_L, and T_A . Thus, the complete model will be a system of four differential equations. This system will be assembled by considering each of the four populations separately. First, the normal T-cells. These cells are produced and die according to the processes identified in the earlier project. In addition, interactions between uninfected T-cells and the virus transfer some normal T cells into latently infected T-cells. Thus, $\frac{dT}{dt} = R_T(t,\ T,\ T_L,\ T_A,\ v)$, where

$$R_T(t,\ T,\ T_L,\ T_A,\ v) = s + r\,T\left(1 - \frac{T + T_L + T_A}{T_{max}}\right) - \mu\,T - k_1\,v\,T.$$

Explain the change in appearance of the numerator of the logistic term.

Next, the latently infected T-cells population is increased as normal T-cells interact with HIV and decrease due to both natural death and by transfer to actively infected T-cells. The differential equation for T_L is therefore $\frac{dT_L}{dt} = R_{T_L}(t,\ T,\ T_L,\ T_A,\ v)$, where

$$R_{T_L}(t,\ T,\ T_L,\ T_A,\ v) = k_1\,v\,T - \mu\,T_L - k_2\,T_L.$$

The actively infected T-cells population grows due to conversion of latently infected T-cells and decreases due to viral replication. If the rate at which viral replication leads to lysis is denoted by b, then the model for these cells is $\frac{dT_A}{dt} = R_{T_A}(t,\ T,\ T_L,\ T_A,\ v)$, where

$$R_{T_A}(t,\ T,\ T_L,\ T_A,\ v) = k_2\,T_L - b\,T_A.$$

The viral population is assumed to grow at a rate proportional to the death rate of the actively infected T cells (with constant of proportionality N). The only ways in which free virions are removed is by combination with uninfected T-cells or by clearance from the body. The equation for the virus will be $\frac{dv}{dt} = R_V(t,\ T,\ T_L,\ T_A,\ v)$, where

$$R_V(t,\ T,\ T_L,\ T_A,\ v) = \ N b T_A - k_1 v T - a v.$$

- **Step III -- Understanding the Parameters and Checking the Model**

The biological interpretations of each of the parameters was explained when it was introduced. However, the units of each parameter have not been discussed. Assuming each of the three T-cell populations and the virus population is measured in cells per cubic millimeter and time is measured in days, determine the units of each parameter.

A realistic numerical value for each of the new parameters is: $b = 0.24,\ \ a = 2.4$, $k_1 = 2.4 \times 10^{-5}$, $k_2 = 3 \times 10^{-3}$, and $N = 1400$.

The appropriateness of the model that has just been constructed would be seriously compromised if any of the populations could become negative (when starting from non-negative initial conditions).

Consider the uninfected T-cell population. The only way this can ever become negative is for $\frac{dT}{dt} < 0$ at some point when $T = 0$. Now, if $T = 0$, then $\frac{dT}{dt} = R_T(t,\ T,\ T_L,\ T_A,\ v) = s$ and we see that $s > 0$ assures that the uninfected T-cell population can never be negative.

Determine conditions which guarantee that none of v, T_L , and T_A can ever be negative.

- **Step IV -- Steady-State Analysis**

The qualitative analysis will focus on two key issues:

- what is the infected steady-state?
- how does an initial infection evolve towards the steady-state solution?

Prior to answering these questions, it is important to understand the importance of these questions. First, it is possible that the onset of AIDS might be avoided if the infected steady-state occurs with a sufficiently high level of uninfected T-cells. The second question is important because even if the steady-state does not provide enough T-cells to balance the virus, it is possible that the evolution to the steady-state would not be completed within the (normal) lifetime of the patient.

Using the given values of the parameters, find all physically realistic steady-state solutions to the system of four ODEs.

One of these steady-state solutions is the steady-state solution for the uninfected case. The other has a reduced T-cell level (well below the normal range) and a fairly high concentration of HIV.

To conclude, let's examine the evolution of the solution to this steady-state solution. Using initial values of $T = 1000$, $T_A = T_L = 0$, and $v = 0.001$ create one plot showing T_A, T_L, and v during the first 150 days of the infection. (Two time steps per day should provide enough resolution to create a reasonable graph.)

Create a separate plot showing the evolution of T . Why is the uninfected T-cell population not included in the original plot?

Note the dramatic initial decrease in the concentration of the free virus. Which term in the equation for v is responsible for this behavior? What is the biological explanation of this?

Note that the solution is very close to the virus-free steady-state solution. While it is possible that the solution will converge to this solution, unfortunately, this does not occur.

Next, continue the calculations until the solution converges to its steady-state solution. When did the uninfected T-cell population first drop below the normal range for a healthy individual? How much longer does it take for the solution to converge to a steady-state solution. Is it likely that this transition will occur during the patient's lifetime?

- **Step V -- Concluding Remarks**

The model developed in this problem is consistent with the current understanding of the transition from infection with HIV to the onset of AIDS. That is, immediately after infection there is an enhanced level of infected T-cells. After only a few days these levels become so low that they are undetectable by the typical testing procedures. During this period, the normal T-cell population is strong enough to combat the virus. However, the level of T-cell production cannot be maintained forever. At this point the T-cell concentration drops to a fraction of the necessary level. AIDS is now very likely to be detected.

While the overall picture presented by this model is rather bleak, there are some bright spots. In particular, this model suggests that the onset of AIDS can be delayed if healthy T-cell production can be extended. This is the idea behind some of the drugs already being tested and used for the “treatment” of AIDS.

- **Bibliography**

[1] Greene,W.C., AIDS and the Immune System, *Scientific American*, September 1993 (Special Issue).

[2] Herod, J.V., A Model for an HIV Infection, *Maple Technical Newsletter*, 10, 1993, p. 72-78.

[3] Herod, J.V., Yeagers, E.K., Mathematics, Biology, and Maple, *Maple Technical Newsletter* (Special Issue), 1994, p. 69-76.

[4] Perelson, A.S., Kerschner D.E., DeBoer R., Dynamics of HIV Infections of $CD4_+$ T-Cells, *Mathematical Biosciences*, **114**,1993, p.81-125.

5.7 Design of an Electrical Network

- **Step I -- Understanding the Problem**

The objective of this project is to design an electrical network to meet specific design criteria. The network consists of two loops. The left loop contains a 12 volt emf and a 1 henry inductor and the right loop contains a 1/4 farad capacitor and 6 ohm resistor. A resistor with an unspecified resistance *r* is on the common side of both circuits (see the accompanying figure). There is no current in the network until a switch, in the left loop, is closed.

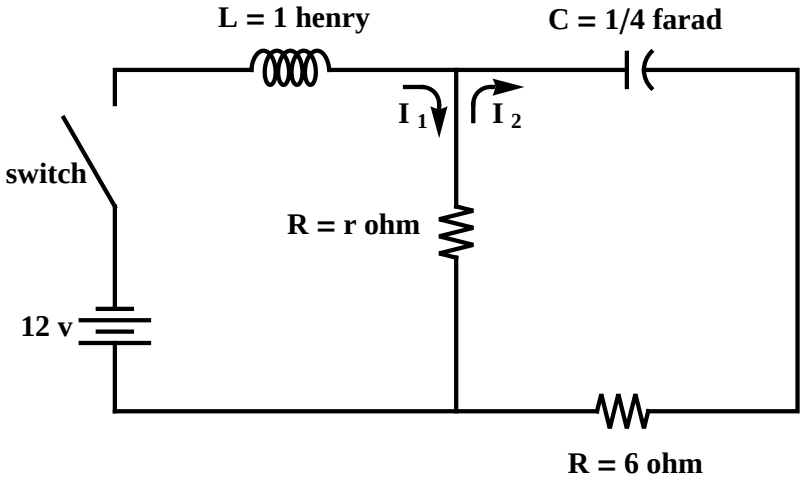


Figure 1: Two-loop network with unspecified common resistance.

The objectives are as follows:

- to determine the steady-state currents (as a function of the unknown resistance, r)
- to identify the different qualitative behavior of the transient currents
- to investigate the length of time required for the currents to converge to within 5% of the steady-state currents

- **Step II -- Mathematical Formulation/Modeling**

Denote the current in the left and right loops by $I_1(t)$and $I_2(t)$, respectively. First-order differential equations for these currents can be obtained by applying Kirchoff's voltage law (Section 3.5, Nagle/Saff/Snider) to each loop of the network.

Write Kirchoff's voltage laws for each loop. Manipulate the equations until they have the (standard) form

$$\begin{pmatrix} I_1'(t) \\ I_2'(t) \end{pmatrix} = A \begin{pmatrix} I_1(t) \\ I_2(t) \end{pmatrix} + b$$

where *A* is a 2×2 matrix and *b* is a two-dimensional vector. (Both *A* and *b* will depend on the parameter *r*.)

(Note that the system is first-order only because neither loop contains both a capacitor and an inductor. How would the equations change if either loop contained both an capacitor and an inductor?)

Since there is no current in either loop prior to the closing of the switch, the initial conditions are $I_1(t) = I_2(t) = 0$.

Step III -- Solution and Analysis

Determine the steady-state solution for all resistances $r > 0$.

Determine, for all $r > 0$, a basis of solutions for the homogeneous equation. Describe the trajectories in the phase plane. (Hint: For what values of r do the solutions oscillate? Does A have a full set of linearly independent eigenvectors for all $r > 0$?)

The phase plane analysis should show that all solutions to the homogeneous equation are transient (for all $r > 0$). This analysis does not give any indication of how long the transient solution takes to become decayed to a point where it is not significant. Determine, as a function of r , the amount of time required to ensure that the current in each loop is within 5% of the steady-state current.

- **Step IV -- Summary and Conclusion**

This network that is considered in this project is not very sophisticated. While the problem would involve more variables and the manipulations would be more difficult for more complicated networks, the same analytical methods could be used to determine answers to the same set of questions.

One additional question that is reasonable to consider is the maximum current that passes through the network. For example, suppose the network is rated to support a maximum current of I_{max} amps. What resistances, r, will guarantee that the current in the network never exceeds this limit?

5.8 Hypergeometric Functions

■ Step I -- Understanding the Problem

The *hypergeometric equation* is the linear second-order ODE

$$x(1-x)y'' + (\gamma - (\alpha+\beta+1)x)y' - \alpha\beta y = 0$$

with parameters α, β, and γ. The Method of Frobenius yields the first solution to this equation (the *hypergeometric functions*):

$$F(\alpha, \beta, \gamma; x) = 1 + \sum_{n=1}^{\infty} \frac{(\alpha)_n (\beta)_n}{n!\,(\gamma)_n} x^n$$

where $(t)_n = \Gamma(t+n)/\Gamma(t)$.

A number of properties of hypergeometric functions were established in the text (Section 8.8 of Nagle/Saff/Snider). For example, $F(1, \beta, \beta; x)$ is a geometric series and $F\left(1/2, 1, 3/2; -x^2\right) = \arctan(x)/x$. In this project, we will develop additional properties of hypergeometric functions.

Benefits derived from the completion of this project will include improved series-manipulation skills and an enhanced knowledge of hypergeometric functions.

■ Step II -- Basic Identities

Verify each of the following facts about hypergeometric functions:

● the power series for F is known to converge on $0 < x < 1$; find the largest set on which the series for F converges.

● determine conditions on α, β, and γ that guarantee that $F(\alpha, \beta, \gamma; x)$ is a polynomial (i.e., when does the series involve only a finite number of non-zero terms?)

● verify that $\frac{d}{dx} F(\alpha, \beta, \gamma; x) = \frac{\alpha\beta}{\gamma} F(1+\alpha, 1+\beta, 1+\gamma; x)$

● express $\frac{d^2}{dx^2} F(\alpha, \beta, \gamma; x)$ in terms of a single evaluation of F (for appropriate arguments) multiplied by an appropriate coefficient

■ Step III -- Hypergeometric and Elementary Functions

Hypergeometric functions can frequently be used to provide a connection between a variety of elementary functions. A number of these relationships will be explored in this part of the project.

● show that $F(\alpha, 1, \alpha; x) = F(1, 1, 1; x) = (1-x)^{-1}$

● show that $F(-n, \beta, \beta; x) = (1-x)^{-n}$

● show that $F(1/2, 1/2, 3/2; x) = \arcsin x$

● express $F(\alpha, \beta, \gamma; 1)$ as a ratio of two products involving the gamma function Γ

Step IV -- Hypergeometric Functions and Differential Equations

The hypergeometric equation is a second-order linear ODE, so it should have two linearly independent solutions. When γ is not an integer, the Method of Frobenius yields a second solution

$$y_2(x) = x^{1-\gamma} F(\alpha + 1 - \gamma, \beta + 1 - \gamma, 2 - \gamma; x)$$

Show that $y_1(x) = F(\alpha, \beta, \gamma; x)$and $y_2(x)$ are linearly independent solutions to the hypergeometric equation.

Derivatives of the hypergeometric function lead to several interesting identities. For example,

$$\frac{d}{dx} F(\alpha, \beta, \gamma; x) = \frac{\alpha \beta}{\gamma} F(1 + \alpha, 1 + \beta, 1 + \gamma; x)$$

Verify this identity. Then, derive the corresponding identities for $\frac{d^2}{dx^2} F(\alpha, \beta, \gamma; x)$ and for any integer $n > 1$, $\frac{d^n}{dx^n} F(\alpha, \beta, \gamma; x)$.

Step V -- Summary and Conclusion

Note that *Mathematica* knows about the hypergeometric function. Actually, **`HypergeometricPFQ`** is the generalized hypergeometric function. This function is similar to *F* , but has a few differences. See the on-line help, **`? HypergeometricPFQ`** for more complete information about hypergeometric functions and *Mathematica*.

While the hypergeometric function is rather complicated, the fact that it is related to so many functions (logarithms, inverse tangent, Legendre polynomials, etc.) gives an indication of the breadth of problems in which hypergeometric functions can arise.

Bibliography

[1] Abrowitz M., Stegun ,I.A. (eds.), *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, Dover, 1972.

Appendix: *Mathematica* Quick Reference

Below is presented quick reference help for essential *Mathematica* commands. A wealth of information is also available on the Web site: http://www.wolfram.com. An extensive list of helpful resources on *Mathematica* and on doing differential equations with *Mathematica* is available by searching this website. At this site are many links to other sites and information about applications to various fields of mathematics, science, and engineering.

■ On-Line Help

Command	Description	Examples
`? Keyword`	display on – line help	`? Plot`

■ Mathematical Operations, Functions, and Constants

Command	Description	Examples
`+, -, *, /, ^`	add, subtract, multiply, divide, power	`(2 / 3) * (x + 2) ^ 3 - 10`
`Sin, Cos, Tan, Cot, Sec, Csc`	trigonometric functions	`Sin[Pi / 3] - Sec[x^2]`
`ArcSin, ArcCos, ArcTan, ArcCot, ArcSec, ArcCsc`	inverse trigonometric functions	`ArcSin[Pi / 4]`
`Exp`	exponential function	`Exp[2 * x]`
`Log`	natural logarithm function	`Log[x^2]`
`Log[10, x]`	common logarithm function	`Log[10, x^2]`
`Abs`	absolute value	`Abs[-4^3]`
`Sqrt`	square root	`Sqrt[48]`
`!`	factorial	`6!`
`Pi` or `π`, `Exp[1]` or `ⅇ`, `I` or `ⅈ`	mathematical constants	`Exp[Pi * I]`
`==, ≠, ≤, ≥, <, >`	equations and inequalities	`a * x^2 + b * x + c == 0` `ⅇ^π > π^ⅇ`

■ *Mathematica* Symbols and Abbreviations

Command	Description	Examples
= , :=	assignment	f = x^2 + 3 f[x_] := x^2 + 3
;	suppress output	Integrate[x^2, x];
/.	substitution	x^2 /. x → 2
->	temporary assignment	x^2 /. x → 2
{}	list delimiter	{1, 2, x, y}
%, %%	references last expression, second last expression, etc.	f = x^2 + 3; Integrate[%, x]
<<	load package	<< Graphics`Graphics`
// N	numeric output	Pi // N

■ General Display, Evaluation, and Manipulation

Command	Description	Examples
Apart	compute partial fraction decomposition	Apart[x / (x^2 - 5 x + 6)]
Clear	clears definition of variable	Clear[x]
Denominator	extracts denominator	Denominator[x / (x^2 - 5 x + 6)]
Numerator	extracts numerator	Numerator[x / (x^2 - 5 x + 6)]
Plot	plot function	Plot[x^2, {x, -4, 4}]
Show	display graphics output	Show[g1, g2]

■ Algebra, Calculus, and Differential Equations

Command	Description	Examples
Collect	collect like powers	Collect[x + 4 y + 5 x y, x]
D	differentiation operator	D[x^2, x]
DSolve	solve ordinary differential equations	DSolve[y'[x] == y[x], y[x], x]
Factor	factor a polynomial	Factor[x^2 - 5 x + 6]
Integrate	definite or indefinite integration	Integrate[x^2, x] Integrate[x^2, {x, 0, 1}]
Limit	calculate limit	Limit[x * Sin[x], x → 0]
NDSolve	solve ODE's numerically	NDSolve[y'[x] == y[x], y[x], {x, 0, 5}]
NSolve	solve polynomial equation numerically	NSolve[x^2 - 5 x + 6 == 0, x]
Simplify	simplify algebraic expressions	Simplify[%]