## *TABLE OF CONTENTS*

# 1. INFORMATION FOR INSTRUCTORS

Many of the exercise sets in *Thomas' Calculus, Tenth Edition* and *Thomas' Calculus, Early Transcendentals, Tenth Edition* contain Computer Algebra System (CAS) exercises, grouped in special sections labeled "Computer Explorations", which can be solved using *Mathematica*. In addition to the CAS exercises, the *Thomas' Calculus* CD-ROM and Web Site contain a collection of 38 *Mathematica* modules designed to help students develop a geometric intuition and deeper understanding and appreciation of calculus concepts, methodologies, and applications. CD/Web site icons mark the locations in the text where material related to these modules is covered.

If you plan on using (or are considering using) some of the CAS exercises or *Mathematica* modules associated with *Thomas' Calculus*, then first spend a few minutes reading through the following information containing general advice and tips for successfully integrating *Mathematica* into your course.

## *A. GETTING STARTED*

### Determine the Computing Needs of Your Students

First, you need to determine *where* students will complete their *Mathematica* assignments. If you plan on having students complete their work on campus, then make sure that *Mathematica* is installed and available in the lab where students will be working.

For those of you who need to purchase *Mathematica* for a campus computer lab, consider an *educational site license*, which will allow your school to install *Mathematica* on several machines at a discounted price. Currently, *Mathematica* runs on most major platforms and operating systems. For more information and technical requirements for the various platforms, visit www.wolfram.com/products/mathematica/platforms/. For more information on the purchasing an academic site license, go to www.wolfram.com/solutions/highered/academicpurchase/.

If *Mathematica* is not available on campus, then you will need to determine if it is feasible for students to purchase their own copies of *Mathematica*. The student version is called *Mathematica for Students* and it is available at greatly reduced price (about 90% off the professional version price). Full-time students and in some cases, part-time students are eligible to purchase the student version. Although it is available online, you may want your campus bookstore to have copies of *Mathematica for Students* available for purchase. For more information about *Mathematica for Students*, go to www.wolfram.com/products/student/mathforstudents/. This site includes information on eligibility requirements for purchasing *Mathematica for Students* along with information about the similarities and differences between the student version and professional version of *Mathematica*.

### Familiarize Yourself With the Student Computing Environment

If your students will be working with *Mathematica* on campus, then you will need to spend some time familiarizing yourself with the computer lab. Be sure you determine how students log onto a computer, start the *Mathematica* program, quit a *Mathematica* session and log out of a computer when a *Mathematica* session is completed. Also find out where students will save their completed electronic files and when the computer lab will be open for student use. Keep track of all this information and pass it on to your students.

### Install *Mathematica* Modules and CAS Exercise Examples

Although students can access all of the *Mathematica* Modules on the CD/Web site, be sure you also make these files available in an electronic folder that students can access in the computer lab. In addition, the electronic files for the CAS Computer Exercise examples contained in this manual should also be placed in an electronic folder that students can access. These files can only be downloaded from the Web site, not the CD-ROM. By making this material available in the lab, students will be able to download any files they need, even if they forget to bring their CD into the lab or if they do not have internet access.

## *B. PLANNING YOUR COURSE*

### Student Background

It will very helpful to collect some academic information about your students and to determine if they have had previous experience working with computers. Knowing a little bit of academic information about your students may help you pick appropriate *Mathematica* modules and CAS exercises to assign to your class. If you plan on having students work in pairs or groups on computer exercises, then you may want to group together students who are less experienced at working with computers with students who have more computer experience. If students will be doing their work in a computer lab and if there are lab assistants working in the lab, you will probably want to provide the lab assistants with some basic information on starting and running *Mathematica*.

### Integrating Computer Exercises into your Syllabus

If assigning CAS exercises and/or *Mathematica* modules is a new experience for you, then I would recommend making the computer exercises worth a relatively small percentage of the total points comprising the course grade. Be very selective in which computer exercises or modules you assign and be sensitive about the amount of time it will take for students to complete the assigned computer labs and exercises.

### Familiarize Yourself with the CAS Exercises and the *Mathematica* Modules

Spend some time working through some of the CAS exercises and the *Mathematica* modules before the semester begins. By doing this, you will get a better feel about which *Mathematica* modules and exercises are most appropriate for your students. If you are new to the *Mathematica* computing environment, I highly recommend completing the first *Mathematica* module, *An Overview of Mathematica*, found on the Web site and/or CD-ROM. This module will introduce you to some of the basic features and commands of *Mathematica*. In addition, read through the CAS Exercise Examples contained in this computer manual and try some of the corresponding exercises in the text to become more comfortable with the *Mathematica* computing environment.

## *C. ASSIGNING MATHEMATICA MODULES AND CAS EXERCISES*

### Do the Exercise Before You Assign Them

Before assigning a CAS exercise or a *Mathematica* module, complete the problem or module yourself. By working through the exercise or module, you can determine if any additional instructions or hints are appropriate to pass on to your students. Another reason for completing the work first is to make sure you are ready to answer any questions your students might have about the assigned computer exercises. It will give you the opportunity to evaluate the length and the level of difficulty of the computer assignment. Sometimes a given CAS exercise or module may look straightforward and easy to complete, but when you try to solve the problem yourself, you might discover it takes more time to complete than you previously estimated.

### Discuss *Mathematica* Assignments in Class

When you are ready to assign a CAS exercise or *Mathematica* module, you may want to give a brief overview of the assigned work in class. Discuss the relevance of the assignment to the course topics you are concurrently covering in class. Neglecting to discuss computer assignments during class may give your students the misconceived idea that the computer assignment is just "busy work" with no real importance to the lecture material and/or written homework assignments.

### Assign Only Portions of Certain Modules

It is not necessary to assign an entire module. You might find it appropriate to assign only portions of some of the modules contained on the CD/Web site. For example, in the module *Take it to the Limit*, the first part of the module only covers Section 1.1 in the text while the remaining parts of the module deal with Section 1.2. Therefore, you might elect to assign only the first part of the module to your students while you are concurrently covering Section 1.1 in the text. You may then decide to assign the remaining parts of the module after covering Section 1.2. There may also be times when you decide to delete parts of a given module to shorten the completion time for students. The key is to customize each module to fit the particular needs of your course.

## Determine if Group Work is Allowed

Determine whether you want students to complete the computer assignments individually or in pairs or groups. As mentioned earlier, one advantage of allowing or encouraging group work is that you can pair up students with weak computer backgrounds with more computer literate students in order to help the less computer literate students adjust to the computer environment. A disadvantage of group work is the danger of having one or two students doing all the work for the rest of the group. Whatever you decide to do, be sure you clearly explain to students whether or not you want them to work together or individually in the computer lab.

## Determine the Acceptable Form for Completed Assignments

Be clear on how you would like students to turn in their work. If you assign one of the modules, you will probably not want them to turn in a hard copy of every page of the completed module. Rather have them to print out a portion of the module. For example, if the module concludes with a couple of *You Try It* exercises for students to complete, then you might have your students only print the last portion of the assignment. Another possibility is have your students turn in their work electronically on a floppy disk or to turn it in as an e-mail attachment. If you decide to have your students turn in their work electronically, be sure to give them clear instructions about the name they should give to their file. For example, if you assign the module entitled *Take it to the Limit*, then a good file name might be something like *smith_lim* indicating the last name of the student (Smith) and a key descriptor of the assignment (lim for limits).

## The Following Section Entitled *Information for Students* is Also For You!

Although the information in the following section is written for students, you should also read the next section to learn more about the *Mathematica* computing environment. If you are already familiar with *Mathematica*, then you might want to quickly skim through the material. However, if you are a new *Mathematica* user, then you may want to read the contents of the following section very carefully.

# 2. INFORMATION FOR STUDENTS

The *Mathematica* modules and the CAS Exercise Examples contained on the CD/Web site were written in files called *Mathematica notebooks*. Information can be entered into a notebook from the keyboard or from special files called *palettes*. The purpose of this section is to discuss notebooks and palettes in more detail and to show how *Mathematica* commands are executed in a notebook.

## *A. MATHEMATICA NOTEBOOKS*

### Cells

A *Mathematica* notebook is a file organized into a sequence of *cells*. Each cell contains a specific type of information that has either been entered by the user or has been created as the result of the execution of a *Mathematica* command. Associated with each cell is a cell bracket in the right margin (see Figure 1).

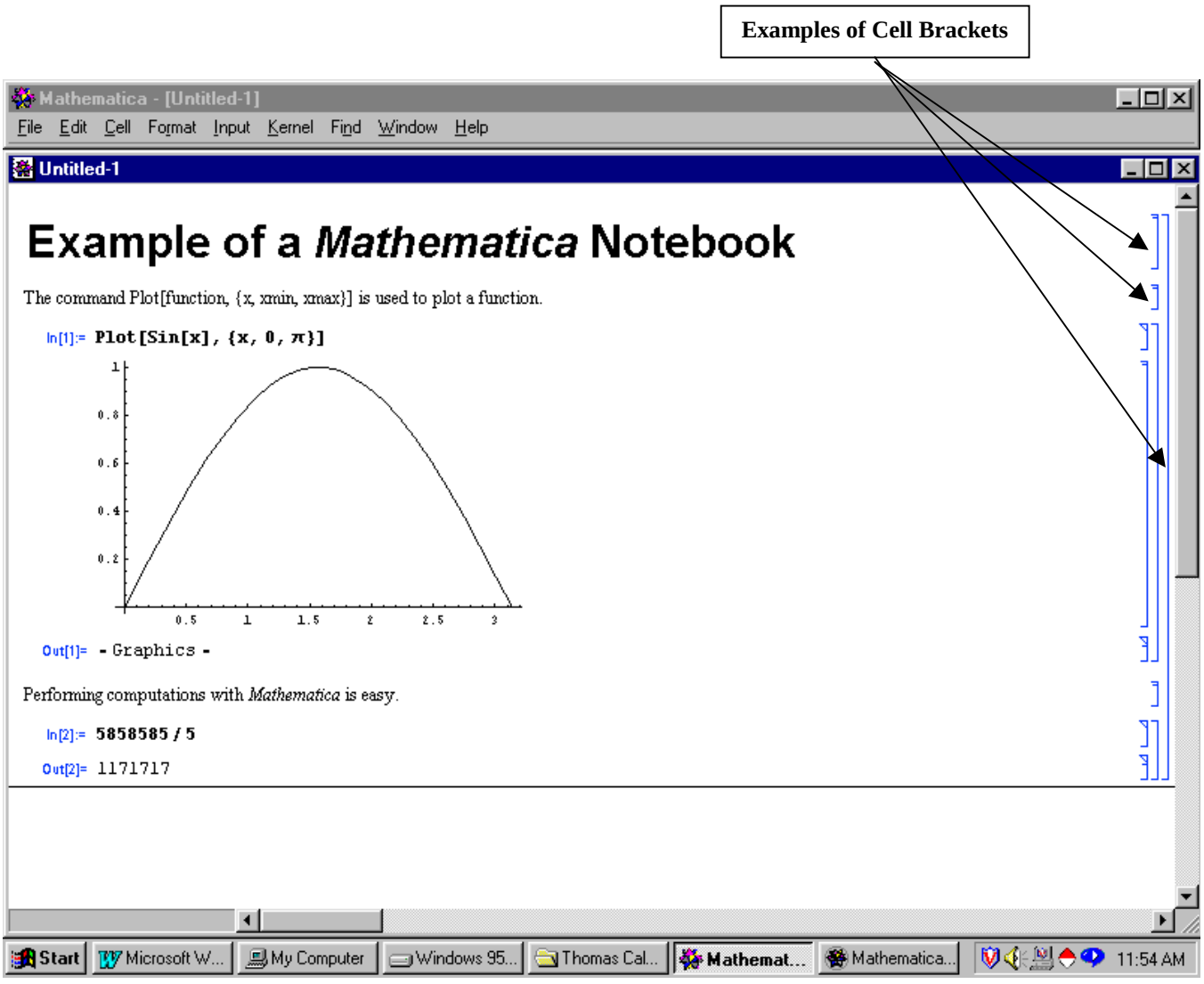



**Figure 1: Example of a *Mathematica* Notebook**

There are a variety of cell types such as *title* and *graphics* cells. Perhaps the three most common types of cells, as shown in Figure 2, are *input* cells, *output* cells and *text* cells. Input cells contain executable *Mathematica* commands. The right bracket of

an input cell contains a little triangle on the upper end of the cell bracket. An output cell contains the results of an input cell that has been executed. A text cell contains information to be read by the user, but contains no executable commands.

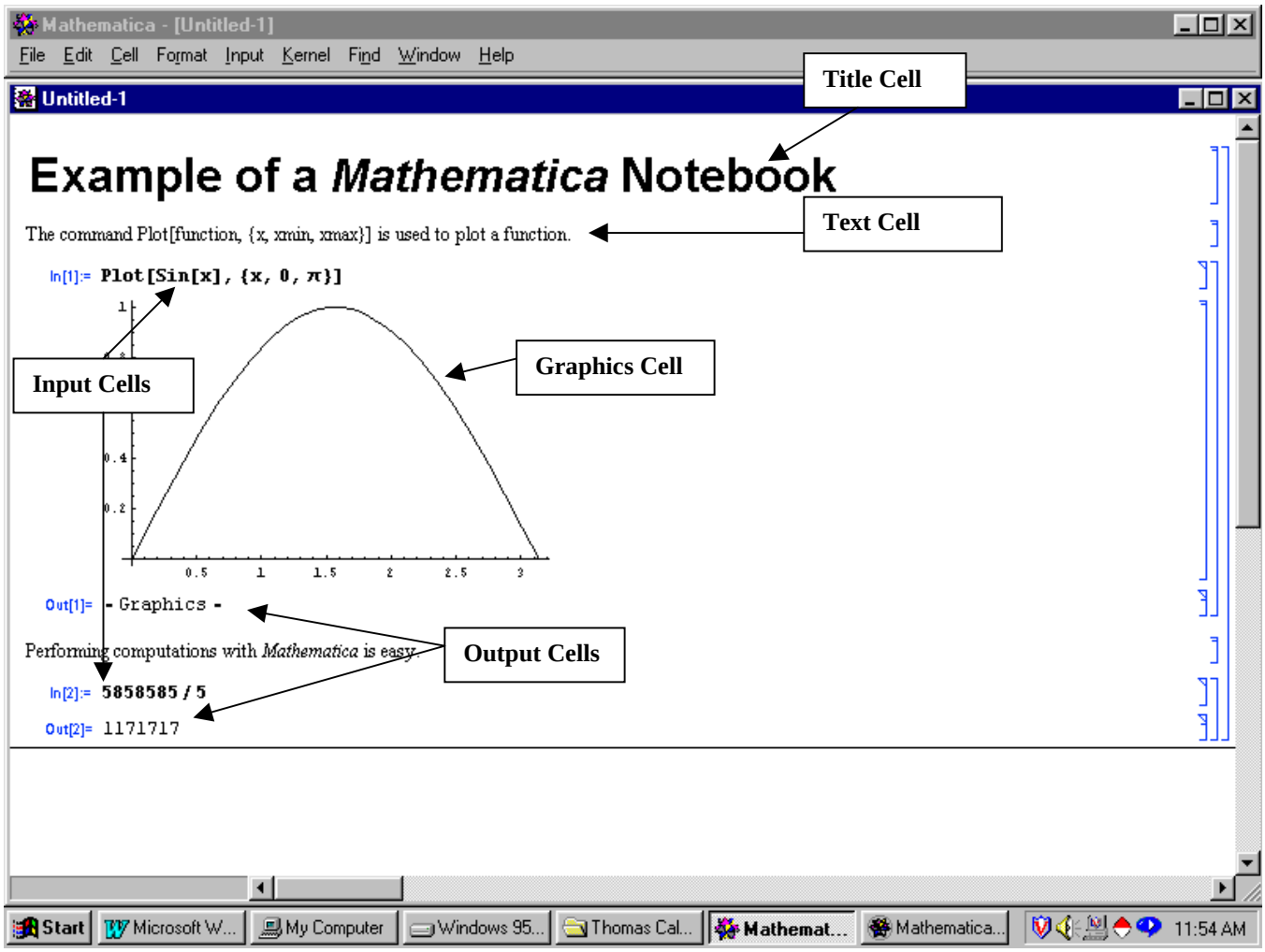



**Figure 2: Example of Cell Types**

## Creating New Cells

The default cell type is an input cell, which means that if you start to enter information into a *Mathematica* notebook, then the input cell type is automatically chosen. If instead, for example, you wish to create a text cell type, then pull down the **Format** menu and select **Style** and then **text** (see Figure 3). To begin a new cell, use the down arrow key or the mouse to click on the location of a new cell. A horizontal line will be displayed and a new cell will replace the horizontal line when you start typing (be sure to use the **Format** menu before typing if you want the cell to be something else other than an input cell).

## Executing *Mathematica* Commands

To execute the contents of an input cell, place the cursor anywhere in the input cell and left-click on the mouse button. Then type **Shift-Enter** or press **Enter** on the numeric key pad on the far right side of the keyboard. When you do this, the word "running" should appear in the upper left-hand corner of the notebook. This indicates that *Mathematica* is processing the contents of the input cell. The output should then appear in an output cell .

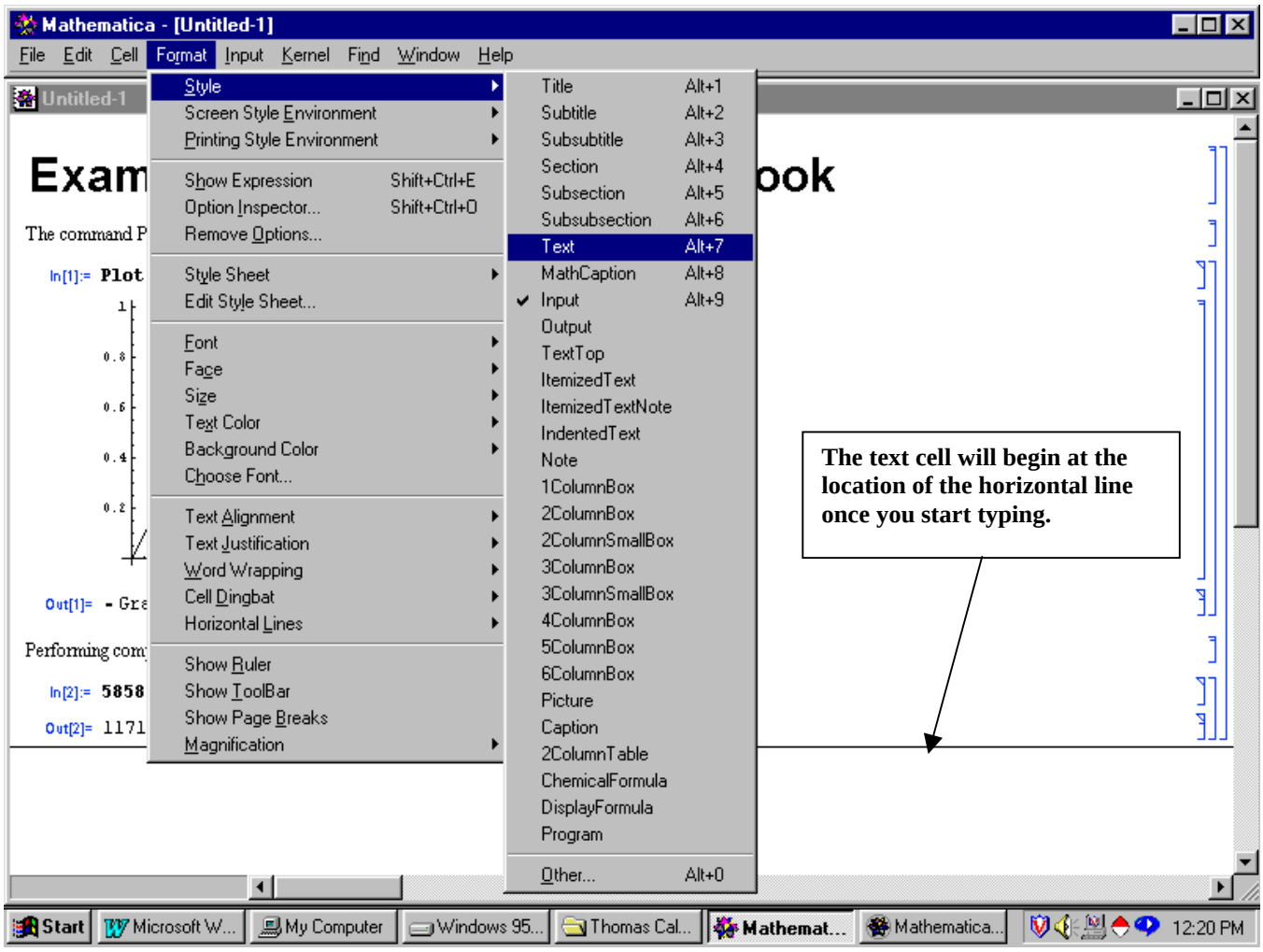



**Figure 3: Selecting a Cell Type**

## Re-executing *Mathematica* Commands

In a notebook, you can always reexecute the contents of an input cell or you can revise the contents of an input cell and then reexecute the cell. Simply click on the cell, make the necessary changes and press **Shift-Enter** to execute the contents of the revised cell.

## Closing and Opening Cell Groups

A group of cells is either *open* or *closed*. When a group of cells is open, all the contents of all the cells are visible to the user. If the group is closed, only the first cell in the group is visible. An upside-down flag indicates a closed group of cells (see Figure 4). You can change a closed group of cells to an open group by double clicking on the upside-down flag. A notebook is grouped in closed cells so that you can see the outline of topics covered.

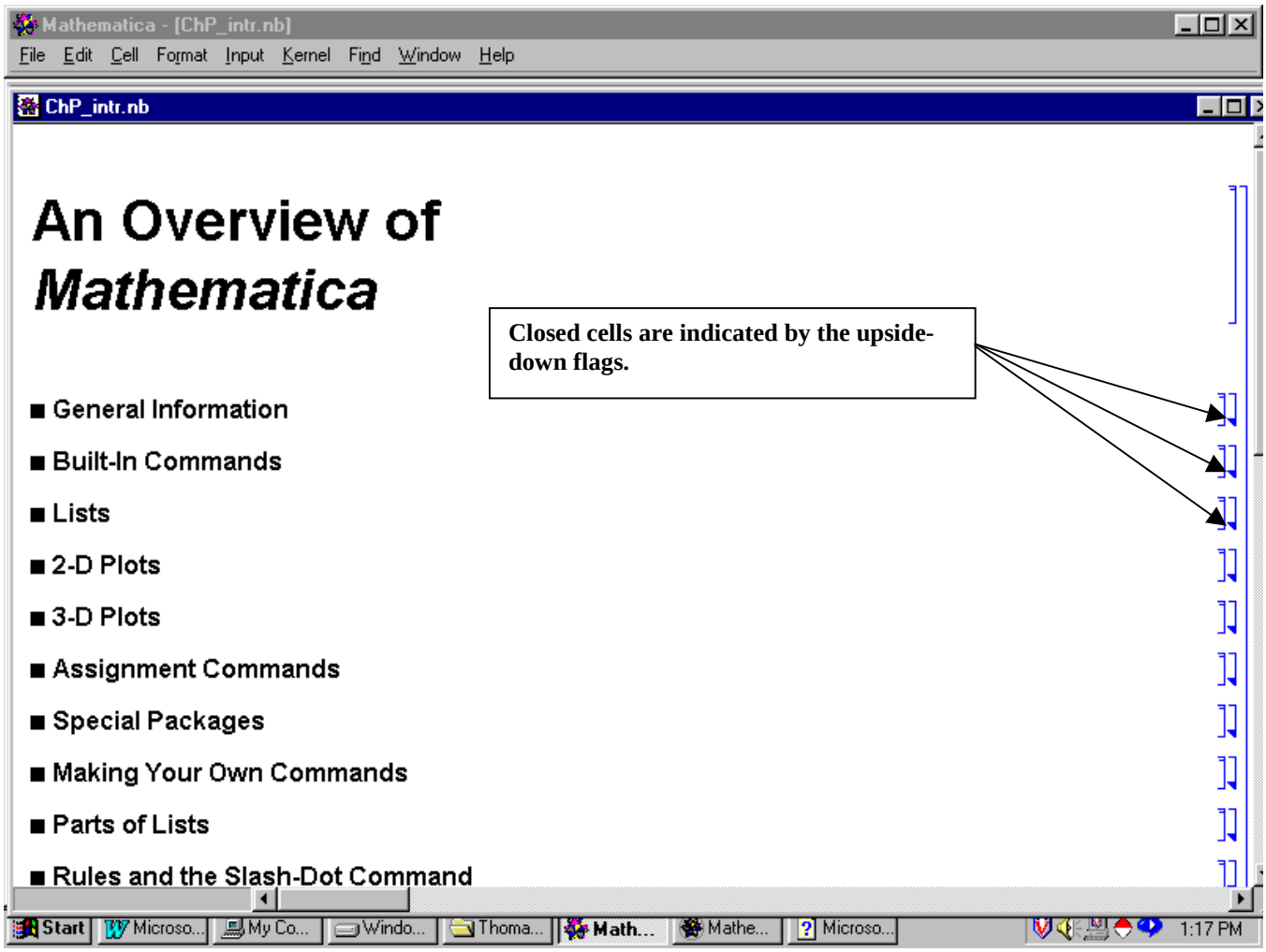



**Figure 4: Examples of Closed Cells in a *Mathematica* Notebook**

## Deleting Cells

To delete an entire cell, simplify click on the cell bracket and press the **Delete** key.

## Palettes

Sometimes you can use *palettes* instead of the keyboard to enter information into a notebook. A palette is similar to a set of calculator buttons, which provide you with shortcuts to entering commands and symbols into a notebook. A palette that you might find useful is **BasicInput** . You can find it by pulling down the **File** menu, then selecting **Palettes** and then selecting **BasicInput**. When the palette appears on the screen, drag it to the right side of the screen and resize the notebook so that they both appear on the screen in non-overlapping windows (see Figure 5). Now, for example, suppose you want to compute the square root of 16. Simply click on the square root button on the palette and the square root symbol will then appear in the notebook. Enter 16 and then type **Shift-Enter** to obtain the result.

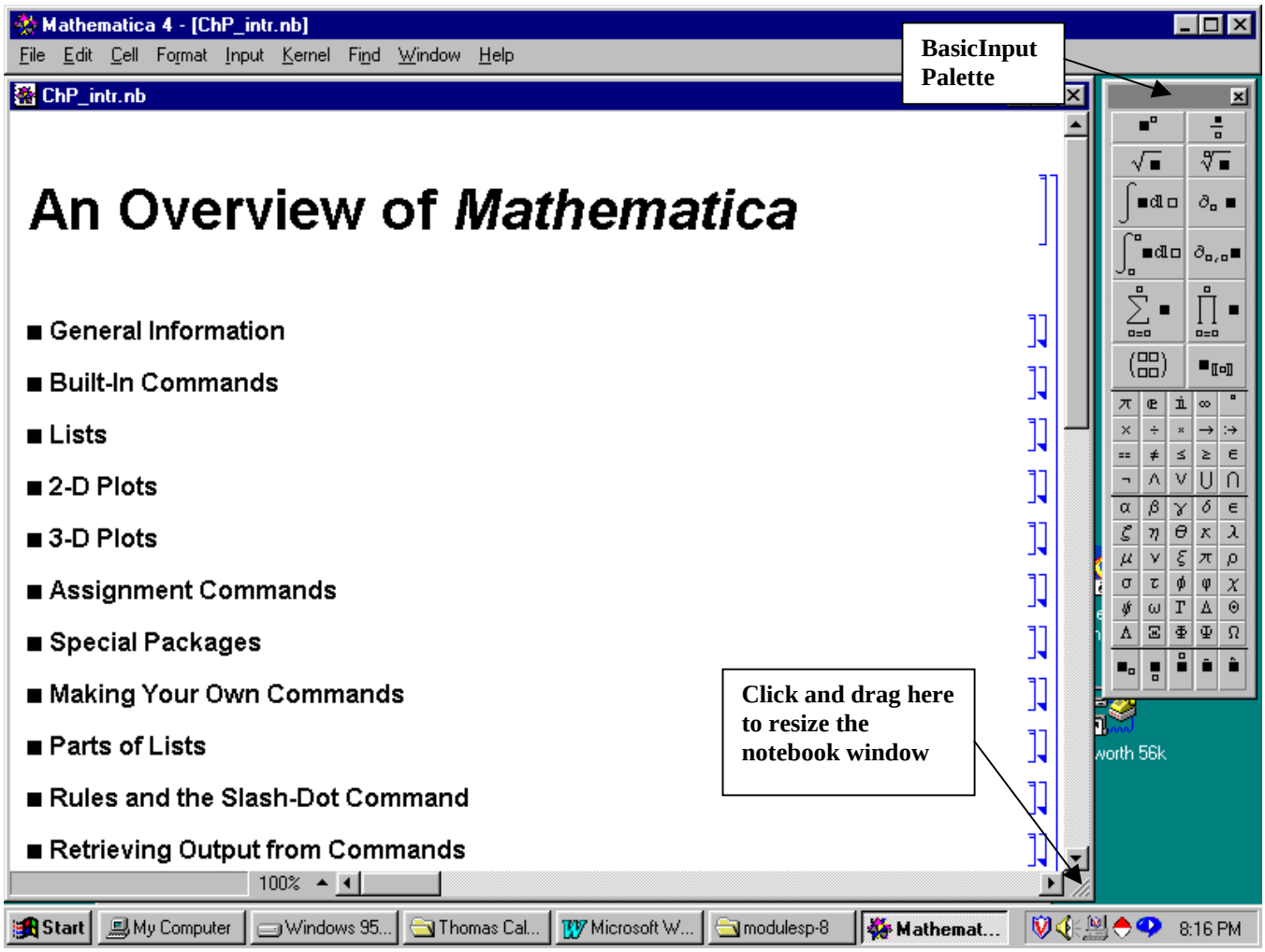



**Figure 5: A *Mathematica* Notebook and a Palette**

## *B. COMMON PROBLEMS ENCOUNTERED*

### Storage Problems

If you plan on saving a *Mathematica* notebook on a floppy disk, make sure the disk has adequate storage space. Attempting to save your work on a floppy lacking an adequate amount of free memory space may result in your entire file becoming corrupted and unusable! If possible, save your work on a hard drive first and then save a backup on a floppy disk. Be sure that the floppy disk has adequate free storage space before saving your work.

### Losing your Work

While working with *Mathematica*, be sure to save your notebook every few minutes to avoid losing your work in case of a power outage or system crash. Nothing is more frustrating that working for a long period of time only to lose your work.

### Syntax Errors

Incorrectly entering a *Mathematica* command will result in a syntax error when the command is executed. An error message will be displayed on the screen as shown in Figure 6. Since *Mathematica* is case-sensitive, make sure you correctly capitalize the appropriate letters in the command before you execute it. When possible, use palettes to enter commands and shortcuts to commands rather than typing commands. More information on correct *Mathematica* syntax can be found later on in this manual in a section entitled *An Introduction to Mathematica* and in the first module, *An Overview of Mathematica*.

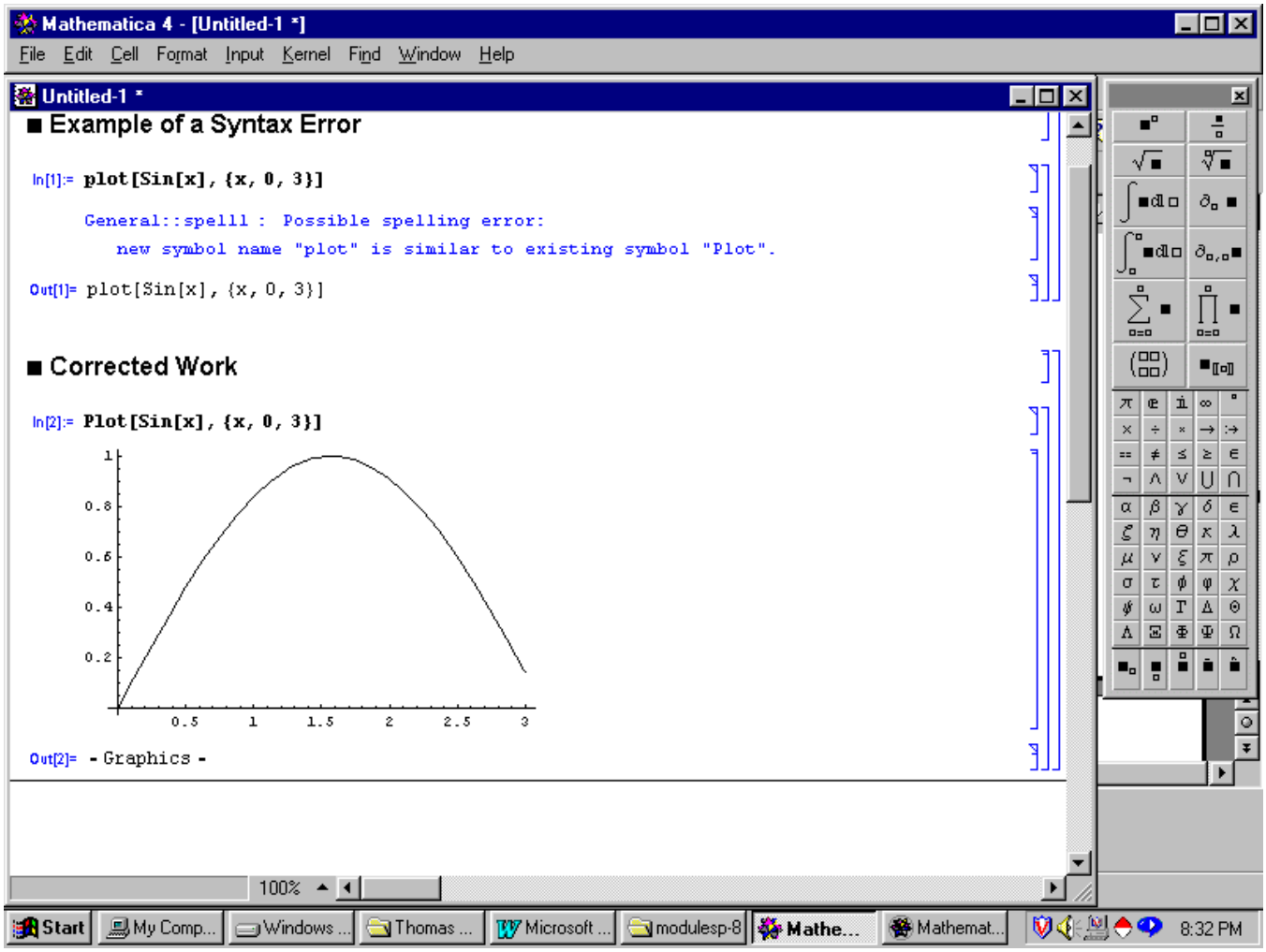



**Figure 6: Example of a Syntax Error and an Error Message**

## Printing problems

A *Mathematica* notebook containing three-dimensional graphics may be very slow to print especially if you are connected to a printer on a local network. To speed of the printing process, hold down the **Ctrl** key and then click on all the brackets containing graphics. Then pull down the **Cell** menu and select **Convert To** followed by **Bitmap**. Converting the graphics images to bitmap images will speed up the printing process.

## *C. MATHEMATICA MODULES*

A list of the 38 *Mathematica* modules contained in the CD/Web site is given below. The title of each module is followed by a set of parenthesis containing the name of the corresponding file on the CD/Web. The objective of each module is given along with prerequisite reading in *Thomas' Calculus* and *Thomas' Calculus, Early Transcendentals, Tenth Edition.*

## 1. An Overview of *Mathematica* (ChP_intr.nb)

The objective of this first module is to introduce you to the basic features, characteristics, and language structure of *Mathematica*. This module should be completed before you begin any of the other modules. It is also strongly recommended that you work through this module before completing any of the CAS exercises in the text.

## 2. Modeling Change: Springs, Driving Safety, Radioactivity, Trees, Fish and Mammals (ChP_mod.nb)

The purpose of this module is to use *Mathematica* to practice a modeling process: consider a behavior, observe data, fit a model, analyze the error, improve the model if appropriate, interpret the model, and make predictions. Prerequisite reading for this module is Section 7 of the Preliminary Chapter.

## 3. Take It to the Limit (Ch1_lim.nb)

The objective of this module is to interpret the limit concept graphically and numerically. Read Section 1.1 before completing Part I of this module and read Section 1.2 before completing the remaining parts of this module.

## 4. Going to Infinity: What Happens to Functions When the Independent Variable Gets Bigger and Bigger and Bigger? (Ch1_inf.nb)

The purpose of this module is to interpret limits going to infinity both graphically and numerically. Read Sections 1.3 and 1.4 to prepare for this module.

## 5. Convergence of Secant Slopes to the Derivative Function (Ch2_sec.nb)

This module will help you visualize the secant line between successive points on a curve, observe what happens as the distance between successive points becomes small, compare the graph of successive secant line slopes with the graph of the derivative function, and apply the definition of the derivative in computing it. Read Sections 2.1 and 2.4 in your text before completing this module.

## 6. Derivatives, Slopes, Tangent Lines, and Making Movies (Ch3_lin.nb)

The objective of this module is to visualize the derivative and the linearization of a function at a point. You will explore the derivative as the slope of a nonlinear function and find the equation of the line tangent to a curve at a point. You will learn how to plot the curve and selected tangents on the same graph. In addition, you will see how to use *Mathematica* to make a movie animation by generating a sequence of plots, each showing a different tangent to the curve. When the sequence of graphs is animated, the tangent lines appear to roll along the graph of the function. Study Sections 2.1, 2.4, 3.6, and 3.7 in your textbook first to prepare for working through this module.

## 7. Motion Along a Straight Line, Part I: Position -> Velocity -> Acceleration (Ch3_mo1.nb)

In this module, you will apply special *Mathematica* functions to analyze position, velocity, and acceleration simultaneously. Three specially designed *Mathematica* commands are introduced here that generate animations to help you visualize the derivative relations among the position, velocity, and acceleration functions. Concepts from Sections 2.2, 2.4, 3.3 are covered in this notebook.

## 8. Newton's Amazing Method: Estimating $\pi$ to *How* Many Places? (Ch3_nm.nb)

The objective of this module is to use *Mathematica* to implement Newton's Method. Read Section 3.7 to prepare for this module.

## 9. Bending of Beams or What Does Calculus Have to Do With the Design of Structures? (Ch4_bb.nb)

In this module, you will learn how engineers use calculus to design structures. Structural engineers need to calculate how beams bend, and they do so by using principles of structural mechanics and calculus. You will investigate some of the ways that engineers use calculus to ensure that the structures they design are both safe and functional.

Before you begin this module, refer to "Maximums, Minimums, and Inflection Points," a Java applet included in the Web/CD. This applet allows you to explore the relationship between the shape of the graph of a function and the values of its first and second derivatives. Read Section 4.1 before completing this module.

## 10. Using Riemann Sums to Estimate Areas, Volumes, and Lengths of Arc (Ch4_avl.nb)

In this module, Riemann sums are used to approximate areas, volumes, and lengths of arc. You will also construct accumulation functions, and see how the accumulated quantities converge to the antiderivative in the limit. Concepts from Sections 4.3, 5.1 and 5.3 are covered in this lab.

## 11. Summing It up with Riemann, Definite Integrals and the Fundamental Theorem of Calculus (Ch4_rs.nb)

The objective of this lab is to visualize the relationship among Riemann sums, the definite integral, and the Fundamental Theorem of the Calculus. You will use each of these concepts to compute or approximate the signed area under the graph of a function representing various applications. Read Sections 4.4 – 4.6, and 6.1 before completing this module.

## 12. Rain Catchers, Elevators, and Rockets(Ch4_rc.nb)

The purpose of this module is to gain an appreciation for the importance of the Fundamental Theorem of Calculus and the Mean Value Theorem in practical applications. You will see the wide applicability of some of the basic ideas in calculus of single variable functions. In addition to gaining a better understanding and appreciation for the Fundamental Theorem of Calculus and the Mean Value Theorem. You will also see the importance of finding the areas between the graphs of functions and identifying extreme values. Before starting, read Sections 4.4 and 4.5 if you have not done so already.

## 13. Motion Along a Straight Line, Part II: Acceleration -> Velocity -> Position (Ch4_mo2.nb)

This module will help you develop an understanding of the integral relationship between acceleration and velocity, and between velocity and position of an object moving along a straight line. Two specially designed *Mathematica* commands are introduced here that animate the integral relations between acceleration and velocity, and between velocity and position of an object moving along a straight line. A variety of motions are studied, including constant velocity, constant acceleration, harmonic oscillation, and decaying oscillations. You can also use the specialized *Mathematica* commands to study other motions that may be of interest to you. Read Section 4.5 before you begin this module.

## 14. Riemann, Trapezoids and Simpson (Ch4_rts.nb)

The objective of this module is to visualize the process of using Riemann sums, the trapezoid rule, and Simpson's rule for approximating definite integrals, and to understand the error associated with each method. Read Section 4.7 before beginning this module.

## 15. Modeling a Bungee Cord Jump: A Classroom Experiment (Ch5_bcj.nb)

In this module, you will be asked to collect data for a bungee cord (or use data collected by Carroll College students); build, test, and refine a model for a bungee jumper; and estimate the length of bungee cord necessary to achieve a desired length of fall. Read Section 5.4 (Section 5.5 in the Early Transcendentals Version) before beginning this module.

## 16. Drug Dosages: Are They Effective? Are they Safe? (Ch6_dd.nb)

The purpose of this module is to use *Mathematica* to solve differential equations related to drug concentrations in the blood when the drug is administered by a single injection, intravenously, or by periodic injections. You will learn how a simple mathematical model and *Mathematica* can be used to regulate the concentration of a prescribed drug in the blood. Read Section 6.4 (Section 5.4 in the Early Transcendentals Version) before beginning this module.

## 17. First-Order Differential Equations and Slope Fields (Ch6_sf.nb)

The purpose of this module is to use *Mathematica* to visualize the slope fields and solution curves for selected first-order differential equations. This module contains a special command that plots slope fields and selected solution curves for first-order differential equations. You can use it to obtain solutions for related problems in the text. In addition, you can use the slope field command to study a wide variety of first-order differential equations and to analyze the long-term behavior of solutions. Read Section 6.6 (Section 5.4 in the Early Transcendentals Version) before beginning this assignment.

## 18. Games of Chance: Exploring the Monte Carlo Technique for Numerical Integration and Computing Probabilities with Improper Integrals (Ch7_int.nb)

The objective of this module is to learn the Monte Carlo method for approximating an integral that cannot be integrated symbolically, and see an application of improper integrals. How can you use a game of chance to evaluate an integral, and what

do games of chance have to do with improper integrals? This module will give you insight into both of these issues. Read Section 7.5 and 7.7 before beginning this module.

## 19. Bouncing Ball (Ch8_bb.nb)

In this module, you will use geometric series to investigate the behavior of a bouncing ball. Properties such as distance traveled and time bouncing will be determined. Read Section 8.3 before beginning this module.

## 20. Taylor Approximations of a Function (Ch8_ts.nb)

Here you will observe an animated demonstration of the convergence of Taylor polynomials to a function that has derivatives of all orders over some interval of its domain. Read Section 8.7 before beginning this module.

## 21. Use the Fourier Series to Approximate Discontinuous Functions and to Interpret Music (Ch8_fs.nb)

The objective of this module is to use *Mathematica* to calculate Fourier series and to build even and odd Fourier representations of selected functions. You will also get a chance to see how Fourier series can be used to build mathematical models of musical tones, to look at their graphs, and even to play back the signal to hear how close our model is to the real thing. Read Sections 8.9 and 8.10 to prepare for this module.

## 22. Using Vectors to Represent Lines and Find Distances (Ch9_lp.nb)

Do you remember how you first learned about the equations of lines in a plane? In this module, you will gain insight into why it is to your advantage to interpret lines in the plane using vectors. Read Sections 9.1 and 9.2 before beginning this module.

## 23. Radar Tracking of a Moving Object (Ch9_rdt.nb)

The objective of this module is to determine how to compute velocity and acceleration of a moving object whose position is given in polar coordinates. Using *Mathematica*, you will animate the trajectory of a moving object together with its position, velocity, and acceleration vectors. Read Sections 9.3 and 9.5 before beginning this module.

## 24. Parametric and Polar Equations with a Figure Skater (Ch9_m2.nb)

Parametric equations are very powerful and the purpose of this module is to help you get used to the idea of expressing curves using parametric equations and analyzing motion in the plane using these parametric equations. Read Sections 9.3 and 9.5 in preparation for this module.

## 25. Putting a Scene in Three Dimensions onto a Two-Dimensional Canvas (Ch10_lt.nb)

The purpose of this module is to use linear transformations to map points from three dimensions onto a two-dimensional space. A portion of this lab focuses on parallel projections that resemble a situation in which the results of X-rays demonstrate the need for a CAT-Scan. Section 10.3 and especially Exercise 61 in Section 10.3 are prerequisite reading for this module.

## 26. Getting Started in Plotting in 3D (Ch10_plt.nb)

In this module, you will learn how *Mathematica* can help you plot the lines, planes, cylinders, and quadric surfaces. In the process of plotting lines and planes, you will gain insight into their vector definitions. Read Sections 10.3 and 10.4 in preparation for this module.

## 27. Moving in Three Dimensions (Ch10_m3.nb)

The objective of this module is to use *Mathematica* to perform calculations to analyze motion in the equations that are given parametrically. Read Sections 10.5 and 10.6 before beginning this module.

## 28. Plotting Surfaces (Ch11_ps.nb)

In this module, learn how to use *Mathematica* to plot surfaces, contours, and level curves. Read Section 11.1 before beginning this module.

## 29. Exploring the Mathematics Behind Skateboarding: Analysis of the Directional Derivative (Ch11_dd.nb)

What is a directional derivative and what does it look like? To find out, this module will allow you get to put yourself in the position of a skateboarder and explore skating on a flat ramp or inside a bowl. The graphical representation of the dot product of the gradient of the surface function and a unit vector in the direction of motion will become clearer. Using *Mathematica*, you will be able to plot the directional derivative as a function of the parameter *t*. Read Section 1l.5 in preparation for completing this module.

## 30. Looking for Patterns and Applying the Method of Least Squares to Real Data (Ch11_ff.nb)

The objective of this module is to minimize the sum of squared residuals to fit an arbitrary function to a set of data. Prerequisite reading for this module is Section 11.7.

## 31. Lagrange Goes Skateboarding: How High Does He Go? (Ch11_lag.nb)

How do you maximize or minimize a function subject to constraints? The skateboarder from the directional derivative project returns, and this time it's Lagrange himself. He will use his multipliers to determine precisely where along the figure-8 he reaches the high and low points of the surface. You will also investigate the role of the directional derivative. Read Section 11.8 before beginning this module.

## 32. How Does Heat Dissipate? (Ch11_heq.nb)

The purpose of this module is to explore the heat equation in order to see physical interpretations of the contours and level curves. The heat equation is a partial differential equation and its solution employs the Fourier series in a meaningful way. Review Fourier series (Section 8.9) and the last part of the additional exercises at the end of Chapter 11 in preparation for this module.

## 33. Take Your Chances: Try the Monte Carlo Technique for Numerical Integration in Three Dimensions (Ch12_mc3.nb)

How can you use a game of chance to evaluate a multiple integral? That is just what you will do with this project. Using *Mathematica*, you will generate random points within a fixed region and then estimate the volume of the desired portion by considering the percentage of random points that fall within the boundaries of the desired portion. Since this will only estimate the exact volume, you will also explore the accuracy of this method. Read Section 12.1 before beginning this module.

## 34. Means and Moments and Exploring New Plotting Techniques (Ch12_mm.nb)

The objective of this module is to extend the concept of the moments of a density function of a solid to applications in probability and engineering. What do means and multiple integrals have to do with probabilities? How can the method of moments be used to help determine whether or not an object will float in an upright position? These questions will be answered as you explore this module. You will also get to practice new and interesting ways to plot functions. Read Sections 12.2 and 12.5 before beginning this module.

## 35. Volumes that You Can Use (Ch12_vol.nb)

The purpose of this module is to use the concept of volume to solve practical problems involving rain catchers and satellite dishes. Read Exercises 31 and 32 in the Additional Exercises at the end of Chapter 12 for background information relevant to this module.

## 36. Work in Conservative and Non-Conservative Force Fields (Ch13_wk.nb)

In this module, you will explore integration over vector fields and experiment with both conservative and non-conservative force functions along different paths. These explorations should help you understand line integrals, as well as better appreciate situations when the work done is independent of the path taken. The background reading for this module is Section 13.3.

## 37. How Can You Visualize Green's Theorem? (Ch13_grn.nb)

In this module, you will explore integration over vector fields and use parameterizations to compute line integrals. You will also explore how to determine the closed curve around which your work integral is a maximum and whether or not it makes any difference if a force is conservative. Read Section 13.4 in preparation for this module.

### 38. Visualizing and Interpreting the Divergence Theorem (Ch13_div.nb)

In this module, you will see that surface integrals are difficult to evaluate, even with the help of *Mathematica*. You will then see that using parameterizations to evaluate flux surface integrals and applying the Divergence Theorem can help with integral evaluations. Read Section 13.8 to prepare for this module.

## *D. COMPUTER ALGEBRA SYSTEM EXERCISES*

### Study the CAS Exercise Examples

Many of the exercise sets in *Thomas' Calculus* contain Computer Algebra System (CAS) exercises, which you can solve with the help of *Mathematica*. The following part of this manual contains examples of CAS exercises found in your text and then solutions to these example exercises are found using *Mathematica*. Before attempting to solve a CAS exercise in your textbook, you will probably find it helpful to first study a relevant example found in this manual.

### Note to those using the Early Transcendentals Version

The following CAS Exercise Examples correspond to the exercise sets found in the regular version of Thomas' Calculus and they correspond to all but three sections of the Early Transcendental version (ET) of Thomas Calculus. CAS examples for Section 2.8 of ET can be found in CAS Exercise Examples for Section 6.2, CAS examples for Section 5.4 of ET are contained in CAS Exercise Examples for Section 6.4 and CAS examples for Section 6.4 of ET are found in Section 6.6 of CAS Exercise Example.

# An Introduction to Mathematica

## ■ Computer Algebra System (CAS) Exercises

In this manual, you will find example exercises, similar to the CAS exercises in your text, and their corresponding solutions, discussed and illustrated with *Mathematica*. To solve a given CAS exercise with *Mathematica*, first study the corresponding example exercise in this manual and then use the example to guide you in solving the CAS exercise in your text.

But before attempting to solve the CAS exercises, spend a few minutes reading through this introduction to familiarize yourself with the basic features, commands and structure of *Mathematica*. You are also strongly encouraged to complete the first *Mathematica* module accompanying the *Thomas' Calculus* text entitled "An Overview of *Mathematica*" contained in the file *ChP_intr.nb*.

## ■ *Mathematica* Arithmetic

You can think of *Mathematica* as a powerful calculator which can do exact as well as approximate arithmetic.

### Addition, Subtraction, Multiplication and Division

The symbols +, − and / are used for adding, subtracting and dividing numbers, respectively. Here are three examples.

```
In[1]:= 575754575849849894 + 748949854985944749598984
Out[1]= 74895043074052059944 8878

In[2]:= 87575750 - 489747598744894574949
Out[2]= -489747598744806999199

In[3]:= 9968686861273254659868650000000000 / 5000
Out[3]= 1993737372254650931973730000000 0
```

The asterisk * or better yet, at least one space, is used to multiply numbers.

```
In[4]:= 6868868686 * 18234987271740
Out[4]= 1252537330604634587336 40

In[5]:= 6868868686 18234987271740
Out[5]= 1252537330604634587336 40
```

### Powers

The ^ stands for the power.

```
In[6]:= 55757^22
Out[6]= 26217227822130734686061732698724649910044114252485611900573633503107377146737
        74557200353459786369112610 49
```

Notice that the last output is such a large number that is fills up several lines! The following input was created by typing 109 **Ctrl-^** 5.

```
In[7]:= 109^5
Out[7]= 15386239549
```

### Palettes

*Mathematica* comes with some standard palettes containing shortcuts to entering commands from the keyboard. A useful palette, called **BasicInput**, contains shortcuts for computing powers, square roots, summations and much more. To open this palette, pull down the **File** menu, choose **Palettes** and then select **BasicInput**. Drag **BasicInput** to the side of your current notebook and if necessary, resize your notebook so that the notebook and palette are not overlapping.

To demonstrate how to use this palette, suppose you want to compute $\sqrt{390625}$. In an **Input** cell, click on the button containing $\sqrt{\square}$. Then enter 390625 and then execute the cell. Your input and output should look like the following.

```
In[8]:= √390625
Out[8]= 625
```

When using any palette key which requires more than one number such as the fraction $\frac{\square}{\square}$ and $\sqrt[\square]{\square}$, use the **Tab** key to move from one number to the next. For example, to compute 757555/5, use the button containing $\frac{\square}{\square}$ in the palette. Enter 757555, then press the **Tab** key and finally enter 5.

```
In[9]:= 757555
        ------
          5
Out[9]= 151511
```

### Exact vs. Approximate Calculations

Study the following input and output statements.

```
In[10]:= √27
Out[10]= 3 √3
```

Notice that *Mathematica* returns an exact answer. This may not always be helpful. For example, the following output is identical to the input since *Mathematica* returns the exact answer in reduced from.

```
In[11]:= 5899/7
Out[11]= 5899/7
```

There are several ways of obtaining an approximate answer as shown in the following cells.

*Mathematica* will display an approximate answer if at least one number in the calculation contains a decimal points.

```
In[12]:= 5899./7
Out[12]= 842.714
```

Placing `//N` immediately after a calculation will produce the same result.

```
In[13]:= 5899/7 // N
Out[13]= 842.714
```

Instead of placing `//N` immediately after a calculation, you can instead place a calculation inside `N[□]`.

```
In[14]:= N[5899/7]
Out[14]= 842.714
```

Entering `N[□,n]` instructs *Mathematica* to attempt to find an n-digit approximation.

```
In[15]:= N[5899/7, 15]
Out[15]= 842.714
```

### Using Previous Results

The percent symbol `%` always represents the last output produced by *Mathematica*.

```
In[16]:= 625/125
Out[16]= 5
```

At this point, 5 is the most recent result produced by *Mathematica*. So the following example will compute $5^2$.

```
In[17]:= %^2
Out[17]= 25
```

The command `%n` represents the result on output line `Out[n]`. For example, the following command will subtract 1 from the result in `Out[15] (verify)`.

```
In[18]:= %15 - 1
Out[18]= 841.714
```

With *Mathematica*, you can perform more than one calculation in a single cell by placing two calculations on separate lines using the **Enter** key.

```
In[19]:= √23.1
         2 - 9
Out[19]= 4.80625
Out[20]= -7
```

## ■ Assigning Names

*Mathematica* has two commands that can be used to assign names to values. One assignment command is called the equals command (=) and the other is the delayed equals command (:=). There is a subtle, yet very important difference between the two commands which will be illustrated shortly.

Suppose we want to assign *x* the value of 2 and *y* the value of 3.

```
In[21]:= x = 2
Out[21]= 2

In[22]:= y = 3
Out[22]= 3
```

The number of letters in a name assigned to a value can be of any length as long as it does not begin with number and as long as there are no spaces between any of the characters in the name.

For example, suppose you want to compute the product of *x* and *y* and assign the name *prod* to the result. Recall that the space appearing between the x and the y represents multiplication.

```
In[23]:= prod = x y
Out[23]= 6
```

If you wish to see the value of *prod* again, just type in the name and execute the cell.

```
In[24]:= prod
Out[24]= 6
```

Now suppose we change the values of *x* and *y*.

```
In[25]:= x = 9
         y = 10
Out[25]= 9
Out[26]= 10
```

Notice that the product does not change!

```
In[27]:= prod
Out[27]= 6
```

The reason the value of *prod* is still 6 is due to the use of the = command. When = is used, the right hand side of the assignment

```
prod=x y
```

is immediately executed and the resulting value is assignment to prod. If instead you had execute the assignment

```
prod:=x y
```

then the right hand side of the assignment statement will not actually be computed until prod appears later on in the input cell or in some other input cell.

The `Clear` command removes assignments that you have made. **It is always a good idea to clear an assigned name before creating a new assignment for that name.**

```
In[28]:= Clear[prod]
```

Now the delayed equals `:=` is used to assign prod to be the product of x and y.

```
In[29]:= prod := x y
```

Since the current value of x and y are 9 and 10, respectively, then executing the following input cell produces a result of 90.

```
In[30]:= prod
Out[30]= 90
```

Now suppose you change the value of x to 4. Then the value of prod changes as expected.

```
In[31]:= x = 4
Out[31]= 4

In[32]:= prod
Out[32]= 40
```

### Suppressing Output

Recall that more than one command can be placed in a single input cell as long as each new command starts on a new line.

```
In[33]:= x = 30
         y = 40
         prod
Out[33]= 30
Out[34]= 40
Out[35]= 1200
```

Suppose you do not want the values of x and y displayed as output. Placing a semicolon (`;`) at the end of a line will suppress the result of the current line from being displayed in the output.

```
In[36]:= x = 90;
         y = 30;
         prod
Out[38]= 2700
```

## ■ *Mathematica* Commands

### Built-In Commands and Constants

*Mathematica* commands consist of a string of letters beginning with a capital letter followed by a series of arguments enclosed in square brackets. Two commands which you have already seen in this chapter are `N` and `Clear`. Here are a few more examples.

If *m* and *n* are integers, `Range[`*m*, *n*`]` produces a list of all the integers from *m* to *n* inclusive.

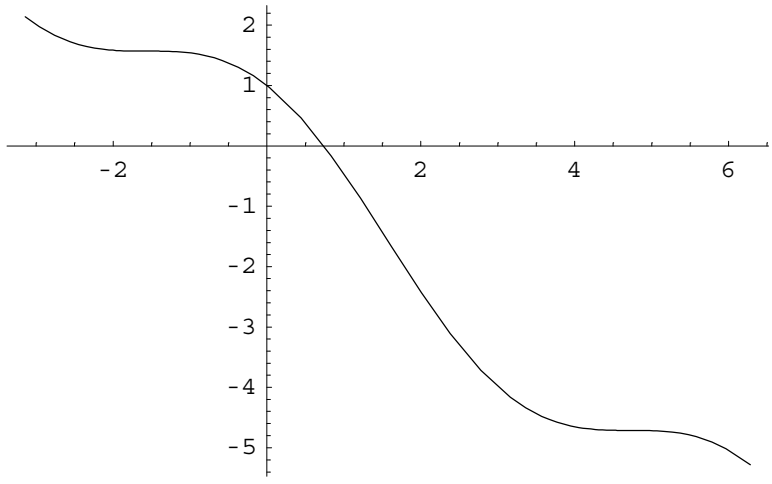
```
In[39]:= Range[1.1, 10]
Out[39]= {1.1, 2.1, 3.1, 4.1, 5.1, 6.1, 7.1, 8.1, 9.1}
```

If *list* represents a list of numbers, then `Min[`*list*`]` will return the smallest number in *list*.

```
In[40]:= Min[%]
Out[40]= 1.1
```

The following `Plot` command will plot the function $f(x) = \cos(x) - x$ for values of $x$ ranging from $-\pi$ to $2\pi$.

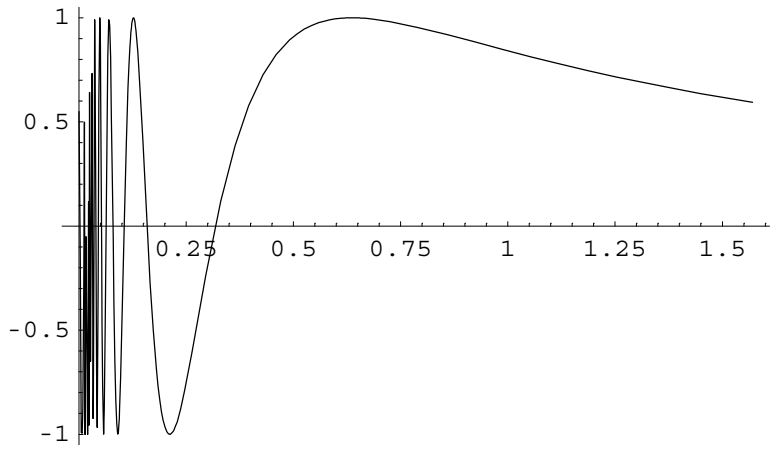
```
In[41]:= Plot[Cos[x] - x, {x, -π, 2 π}]
```




```
Out[41]= - Graphics -
```

Placing a semicolon after a command will suppress any output from being displayed. In studying the last output, the actual output was not the graph itself, but rather the word `-Graphics-`. So placing a semicolon at the end of a plot command will only suppress `-Graphics-` from being displayed.

In[42]:=
```
Plot[Sin[1/x], {x, 0, π/2}];
```




In *Mathematica*, an equation such as $\cos(x) - x = 0$ is represented in *Mathematica* by `Cos[x]-x==0`. Since the cosine function is a *Mathematica* command, it must begin with a capital letter and also note that a double equals sign (==) is used in a *Mathematica* equation (a single equals sign represents an assignment command and therefore cannot be used). The following `FindRoot` command locates a solution to cos(x)-x=0 near $x = 1$.

In[43]:=
```
FindRoot[Cos[x] - x == 0, {x, 0, 2}]
```

Out[43]=
```
{90 → 0.739085}
```

Notice that the `FindRoot` command consists of two words (`Find` and `Root`) which are both capitalized. Notice there is no space separating the two words. *Mathematica* will always capitalize the first letter in each word and will never leave any spaces between the words in a single command.

Some *Mathematica* commands represent constants. For example the command `Pi` is $\pi$ and `E` is Euler's number $e$. The following commands will display their approximate values.

In[44]:=
```
N[Pi]
N[E]
```

Out[44]=
```
3.14159
```

Out[45]=
```
2.71828
```

Rather than typing in commands from the keyboard, some commands can be entered from a standard palette. For example, in the palette **BasicInput**, Pi is represented by $\pi$ and E is represented by $e$.

In[46]:=
```
N[π]
N[e]
```

Out[46]=
```
3.14159
```

Out[47]=
```
2.71828
```

As another example, the command for computing the square root of a number $x$ is `Sqrt[x]`.

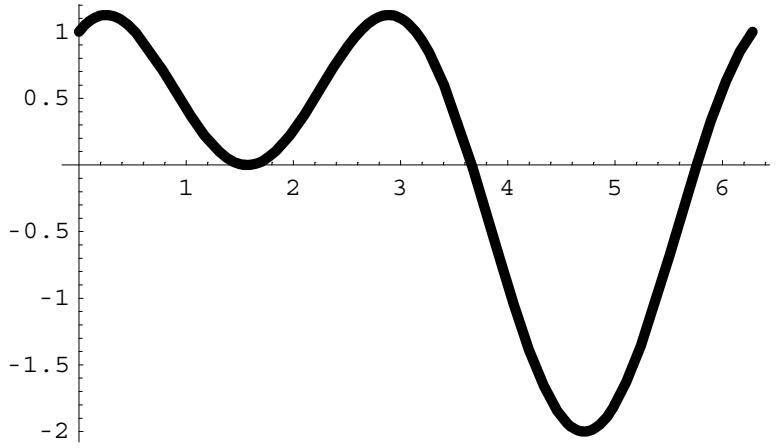
```
In[48]:= Sqrt[16]
Out[48]= 4
```

But as seen earlier, a more natural way of computing square roots is the use the button containing $\sqrt{\square}$ in **BasicInput**.

### Command Options

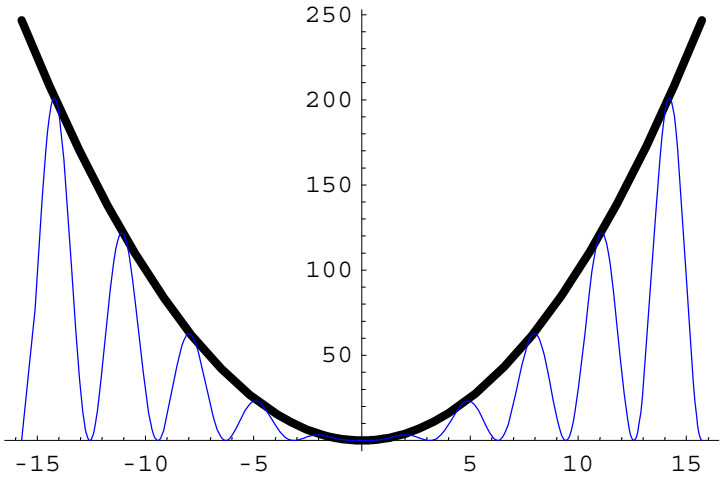
Many *Mathematica* functions, especially the commands which produce graphs, contain options for enhancing the output. For example, the option `PlotStyle → Thickness[.015]` adjusts the thickness of the curve being plotted.

```
In[49]:= Plot[Sin[x] + Cos[2 x], {x, 0, 2 π}, PlotStyle → Thickness[.015]];
```




In the following `Plot` command, two functions contained in brackets{ ...}, $x^2$ and $x^2 \sin^2 x$ are plotted simultaneously with an option for changing the thickness of the first curve and the color of the second curve.

```
In[50]:= Plot[{x^2, x^2 Sin[x]^2}, {x, -5 π, 5 π},
          PlotStyle → {Thickness[.012], RGBColor[0, 0, 1]}];
```



### Help Menu

To see information about a given *Mathematica* command, enter a ? followed immediately by the name of the command.

```
In[51]:= ? FindRoot

         FindRoot[lhs==rhs, {x, x0}] searches for a numerical
           solution to the equation lhs==rhs, starting with x=x0.
```

For more detailed information use ??.

```
In[52]:= ?? FindRoot

         FindRoot[lhs==rhs, {x, x0}] searches for a numerical
           solution to the equation lhs==rhs, starting with x=x0.

         Attributes[FindRoot] = {HoldAll, Protected}

         Options[FindRoot] = {AccuracyGoal → Automatic, Compiled → True, DampingFactor → 1,
           Jacobian → Automatic, MaxIterations → 15, WorkingPrecision → 16}
```

Another excellent source of information is the **Help** menu. You are encouraged to explore the **Help** menu to learn more about *Mathematica* and its features.

### Standard Add-On Functions

In addition to the standard *Mathematica* functions which are available to you as soon as you begin a *Mathematica* session, there are over 1000 add-on functions contained in packages which must first be loaded into computer memory before being used. For example, in a file named `Graphics`, there is a package called `ImplicitPlot` containing a function called `ImplicitPlot`. To use this function, you must first execute the command `<<Graphics`ImplicitPlot`` to load the package into computer memory.

```
In[53]:= << Graphics`ImplicitPlot`
```

Once the package has been loaded into memory, the command `ImplicitPlot` is now defined and ready for use. It will plot the graph of an equation as illustrated below. The `PlotPoints→30` option is used to make the plot the curve look less choppy and the `AspectRatio -> 2.4/1.6` option changes the height to width ratio of the plot.

```
In[54]:= ImplicitPlot[x^2 + y^4 == y^2 - x, {x, -1.3, .3}, {y, -1.2, 1.2},
           PlotPoints -> 30, AspectRatio -> 2.4/1.6, PlotStyle -> RGBColor[0, 0, 1]];
```

### Creating Functions

You can create your own *Mathematica* commands. Since *Mathematica* commands always begin with a capital letter, you should get in the habit of beginning the names of your commands with a lower case letter. The name of your command can be a string of letters followed immediately by square brackets containing the variables in the function. An underscore _ is placed immediately after each variable on the left-hand side of the assignment statement to alert *Mathematica* to the fact that the a variable has been declared. The assignment statement used here is =, but a := could have also been used. (Sometimes the delayed equals := will be preferred over the =.) The following lines of input begin with a clear command since the variables used were assigned specific values earlier in this notebook.

```
In[55]:= Clear[x, y];
         f[x_] = x^2 Sin[x]^2;
```

The variable $x$ is called a dummy variable since it can be replaced by any other variable or number.

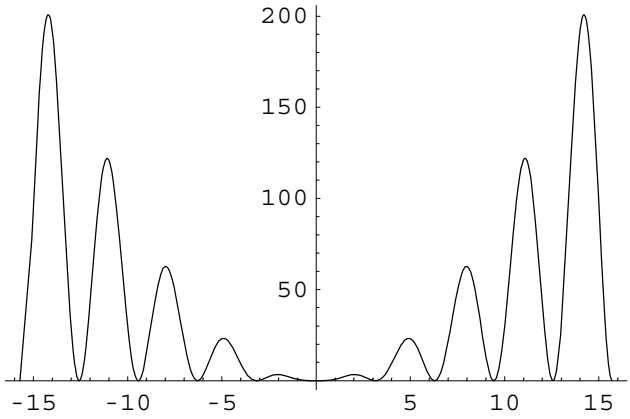
```
In[57]:= f[y]
Out[57]= y^2 Sin[y]^2
```

Recall that variables can be a string of letters. So in the following input, the dummy variable $x$ is replaced with *fred* and so every occurrence of $x$ in the function is replaced with *fred*.

```
In[58]:= f[fred]
Out[58]= fred^2 Sin[fred]^2
```

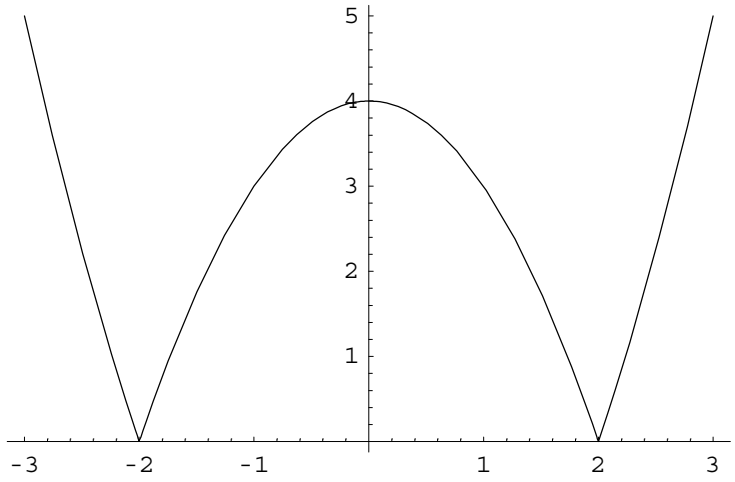
You can also evaluate the function for a given value such as $x = \frac{\pi}{2}$ and you can plot a function you created.

```
In[59]:= f[π / 2]
         Plot[f[x], {x, -5 π, 5 π}];
Out[59]= π^2/4
```




Before redefining a function, it is always a good idea use the `Clear` command to clear out the current definition of the function. In the following, the function `f` is redefined to equal $|x^2 - 4|$. `Abs` is the absolute value command.

```
In[61]:= Clear[f];
         f[x_] = Abs[x^2 - 4];
         Plot[f[x], {x, -3, 3}];
```




## ■ Lists

### Definition of a List

One of the most important structures in *Mathematica* is a list. A list is an ordered array of elements contained inside braces { ...}. Here is an example of a list of numbers.

```
In[64]:= {1, 2, 3, 4, 5}
Out[64]= {1, 2, 3, 4, 5}
```

Notice that the elements of the list are separated by commas. As the following example shows, a list does not need to just contain numbers.

```
In[65]:= {x, {1, 2}, π}
Out[65]= {x, {1, 2}, π}
```

The list just created contains three elements: an unassigned variable x, a list {1, 2} and the number $\pi$.

### Creating Lists

A list can be created and assigned a name.

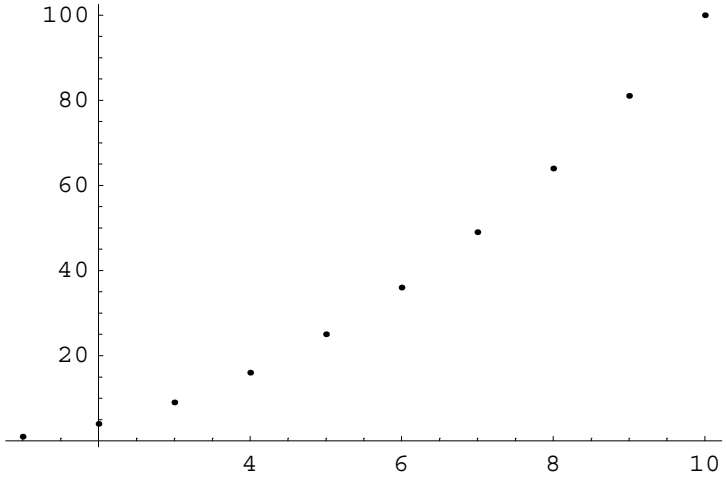
```
In[66]:= first4 = {1, 2, 3, 4}
Out[66]= {1, 2, 3, 4}
```

The `Table` command can be used to create a list. For example, suppose we want to create a list containing the first 15 positive integers. The command `Table[a[i],{i,1,n}]` will create a list of the form `{a[1], a[2], ..., a[n]}`. Here are couple of examples.

```
In[68]:= Table[i, {i, 1, 15}]
Out[68]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}

In[69]:= pts = Table[x^2, {x, 1, 10}]
Out[69]= {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

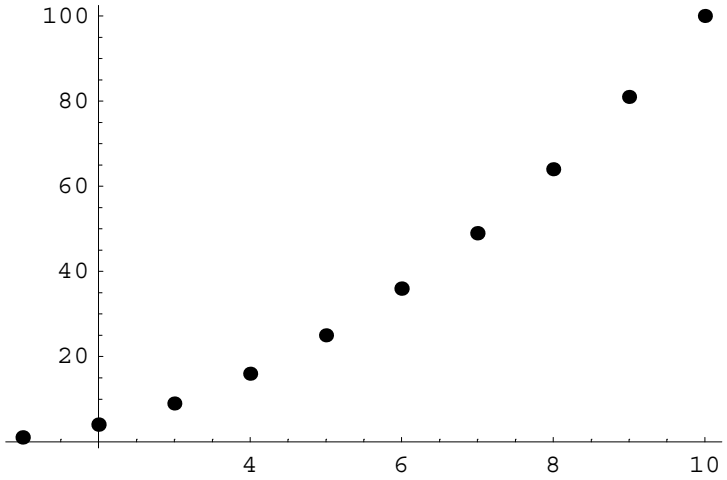
Given a list $\{y_1,\ y_2,\ y_3,\ \ldots,\ y_n\}$, the `ListPlot` command can be used to plot the points $(1,\ y_1),\ (2,\ y_2),\ \ldots,\ (n,\ y_n)$.

```
In[70]:= ListPlot[pts];
```




The points are difficult to see, so an option is added to make the points for visible.

```
In[71]:= ListPlot[pts, PlotStyle -> PointSize[.02]];
```

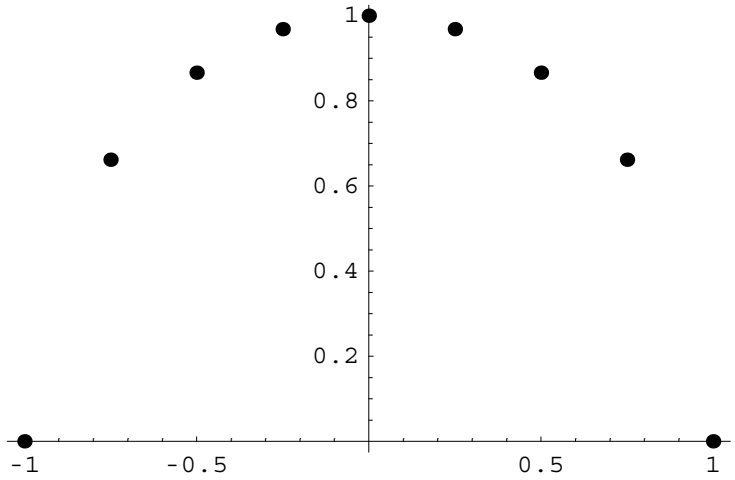



Next, the `Table` command is used to create a list of lists, each containing two numbers.

```
In[72]:= newpts = Table[{x, Sqrt[1 - x^2]}, {x, -1, 1, .25}]
Out[72]= {{-1, 0}, {-0.75, 0.661438}, {-0.5, 0.866025}, {-0.25, 0.968246},
          {0., 1.}, {0.25, 0.968246}, {0.5, 0.866025}, {0.75, 0.661438}, {1., 0.}}
```

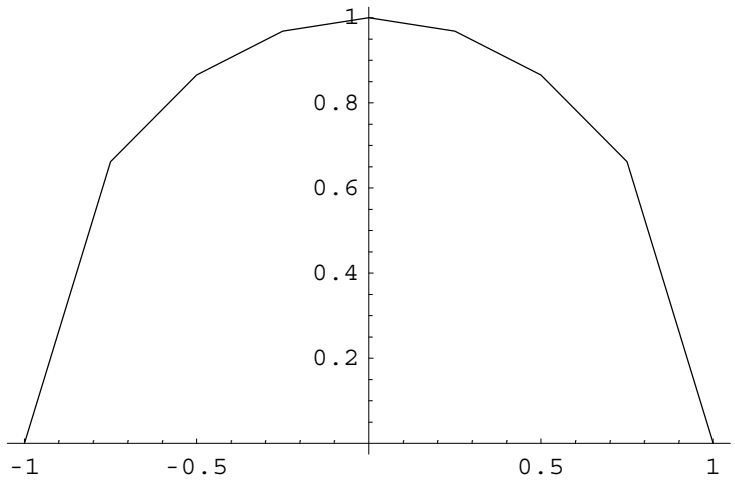
In the list *newpts*, each list of two points can be thought of as an ordered pair which can then be plotted.

```
In[73]:= ListPlot[newpts, PlotStyle -> PointSize[.02]];
```




The following PlotJoined→True option will connect the points.

```
In[74]:= ListPlot[newpts, PlotJoined -> True];
```




### Extracting Elements of a List

Consider the list created earlier called *pts*.

```
In[75]:= pts
Out[75]= {1, 4, 9, 16, 25, 36, 49, 64, 81, 100}
```

To extract the ith element in the list, use the command pts[[i]].

```
In[76]:= pts[[9]]
Out[76]= 81
```

Recall that the list newpts consists of a list of lists of length 2.

```
In[77]:= newpts
Out[77]= {{-1, 0}, {-0.75, 0.661438}, {-0.5, 0.866025}, {-0.25, 0.968246},
          {0., 1.}, {0.25, 0.968246}, {0.5, 0.866025}, {0.75, 0.661438}, {1., 0.}}
```

Executing `newpts[[i]]` will extract the ith element of the list which itself is a list.

```
In[78]:= newpts[[2]]
Out[78]= {-0.75, 0.661438}
```

Executing `newpts[[i,j]]` will return the jth element in the ith list. For example, the following command extracts the 2nd element in the fourth list contained in `newpts`.

```
In[79]:= newpts[[4, 2]]
Out[79]= 0.968246
```

Here is another example where the first element in the last is extracted.

```
In[80]:= newpts[[9, 1]]
Out[80]= 1.
```

### Extracting Solutions to an Equation

When you use *Mathematica* to find the solution to an equation, the output will be displayed in list form and therefore the solutions can be extracted using the ideas discussed previously in this introduction.

Consider the command `FindRoot[eqn, {x, x0}]` which attempts to find a solution to the equation *eqn* near $x_0$. For example, suppose we are searching for the positive solution to $x^2 - 3 = 0$. In the following input statement, the result of the `FindRoot` command is assigned the name sol.

```
In[81]:= sol = FindRoot[x^2 - 3 == 0, {x, 2}]
Out[81]= {x → 1.73205}
```

Notice that the output is a list containing `x→1.73205` and this is the only element in the list.

```
In[82]:= sol[[1]]
Out[82]= x → 1.73205
```

The single element in the list can be divided into two parts. The first part is *x* and the second part is 1.73205.

```
In[83]:= sol[[1, 1]]
Out[83]= x

In[84]:= sol[[1, 2]]
Out[84]= 1.73205
```

Therefore the actual solution is extracted using `sol[[1,2]]`.

Now consider the following command which finds all the solutions to a given polynomial equation.

```
In[85]:= polysol = NSolve[x^4 - 3 x^2 + x == 0, x]
Out[85]= {{x → -1.87939}, {x → 0.}, {x → 0.347296}, {x → 1.53209}}
```

The solution consists of a list of 4 lists. For example, to extract the third list, use the following command.

```
In[86]:= polysol[[3]]
Out[86]= {x → 0.347296}
```

The third element is also a list containing one element, `x→0.347296 which is extracted as follows.`

```
In[87]:= polysol[[3, 1]]
Out[87]= x → 0.347296
```

The actual third solution is the second part of `x→0.347296 which can therefore be obtained by executing the following command.`

```
In[88]:= polysol[[3, 1, 2]]
Out[88]= 0.347296
```

Here is how you can extract the last solution in `polysol`.

```
In[89]:= polysol[[4, 1, 2]]
Out[89]= 1.53209
```

## ■ Creating Animations

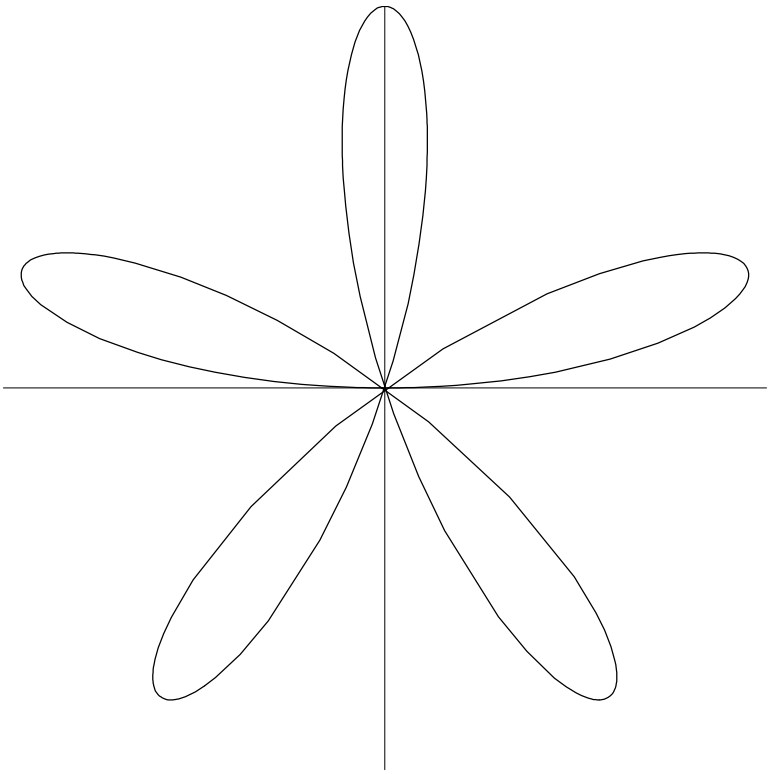
The `Table` command used to form lists can also be used to form a list of frames in a graphics movie which can then be animated with *Mathematica*.

To illustrate how to create a movie, the add-on command `PolarPlot` contained in the `Graphics` package will be used.

```
In[90]:= << Graphics`Graphics`
```

Suppose you want to graph the equation $r = \cos(5\,\theta)$ in the polar coordinate system. The following input will do this where the option `PlotRange->{{-1,1},{-1,1}}` instructs *Mathematica* to plot the graph with the horizontal axis ranging from $-1$ to 1 and the vertical axes ranging from $-1$ to 1. The other option given here is Ticks->None which will leave the axes unlabeled.

In[91]:=
```
PolarPlot[Sin[5 θ], {θ, 0, π}, PlotRange -> {{-1, 1}, {-1, 1}}, Ticks -> None];
```



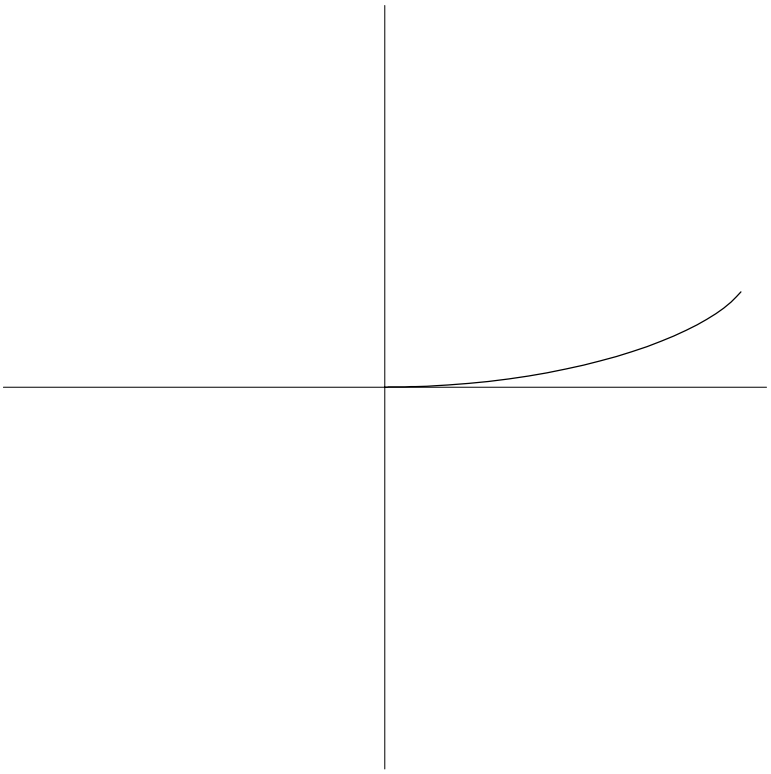
The following `Table` command will form a list of the graphs

`PolarPlot[Sin[5 θ], {θ, 0, θ0},PlotRange->{{-1,1},{-1,1}}, Ticks->None];`

for $\theta_0 = \frac{\pi}{12}, \frac{2\pi}{12}, \frac{3\pi}{12}, \ldots, \pi$ (only the first frame is shown to save space).

To animate the created frames, double click on top of any one of the given frames.

In[92]:=
```
polarframes = Table[PolarPlot[Sin[5 θ], {θ, 0, θ0},
    PlotRange -> {{-1, 1}, {-1, 1}}, Ticks -> None], {θ0, π/12, π, π/12}];
```

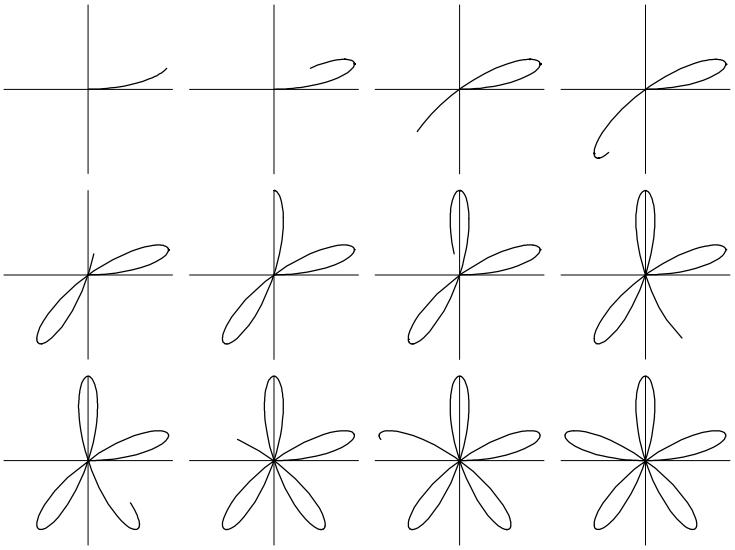


If you would rather just see the frames all at once on the screen, we can take the list of 12 frames just created and divide the list into a list of 3 lists using the `Partition` command.

```
In[93]:= framesdisplay = Partition[polarframes, 4]

Out[93]= {{-Graphics-, -Graphics-, -Graphics-, -Graphics-},
          {-Graphics-, -Graphics-, -Graphics-, -Graphics-},
          {-Graphics-, -Graphics-, -Graphics-, -Graphics-}}
```

Then execute the following command.

In[94]:=
```
Show[GraphicsArray[framesdisplay]];
```



For a more sophisticated example, suppose you want to create a movie illustrating the relationship between the unit circle and the sine curve. The purpose of the following example is to show you the power of animation, so don't worry about understanding all of the details.

First, a plot of $y = \sin(x)$ is created and given the name `sincurve`. The option `DisplayFunction->Identity` will suppress the graph from being displayed for now and the option `AspectRatio -> 2/(2 π+2)` produces a graph whose ratio of height to width is 2 to $2\pi + 2$. The other options have been described previously in this chapter.

In[95]:=
```
sinecurve = Plot[Sin[x], {x, 0, 2 π},
    PlotRange -> {{-2, 2 π}, {-1, 1}}, DisplayFunction -> Identity,
    AspectRatio -> 2/(2 π + 2), PlotStyle -> RGBColor[1, 0, 0]];
```

Next, a unit circle is created centered at $(-1, 0)$ with a radius of 1 with the `Circle` command. The `Graphics` command is added so that the graph of the circle can be plotted later on with the `Show` command.

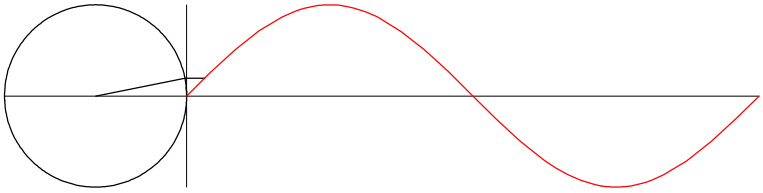
In[96]:=
```
cir = Graphics[Circle[{-1, 0}, 1], PlotRange -> {{-2, 2 π}, {-1, 1}}];
```

In the following input cell, a list is created where each element in the list is a plot of the points $(-1, 0)$, $(\cos(t) - 1, \sin(t))$, $(t, \sin(t))$ are joined together, for $t = \frac{\pi}{16}, \frac{2\pi}{16}, \frac{3\pi}{16}, \ldots, 2\pi$, using the `ListPlot` command described earlier in this chapter.

In[97]:=
```
cirptslines = Table[ListPlot[{{-1, 0}, {Cos[t] - 1, Sin[t]}, {t, Sin[t]}},
    PlotRange -> {{-2, 2 π}, {-1, 1}}, DisplayFunction -> Identity,
    AspectRatio -> 2/(2 π + 2), Ticks -> None, PlotJoined -> True], {t, π/16, 2 π, π/16}];
```

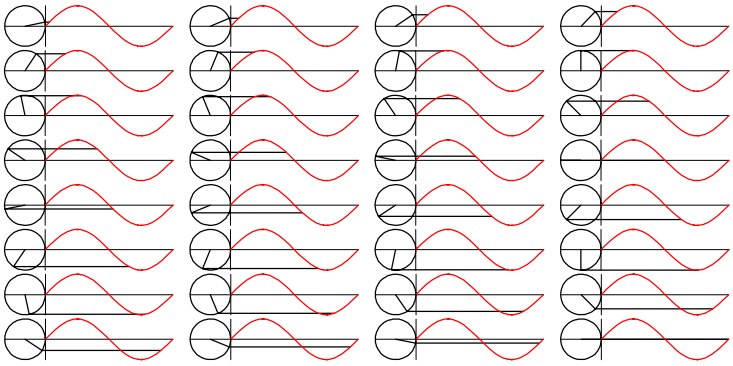
Now the `Table` command is used to show the frames of the movie. Since the option `DisplayFunction->Identity` was used in the previous commands to suppress the graphics output, the command `DisplayFunction->$DisplayFunction` must now be used to instruct *Mathematica* to show the graphics previously hidden. (Only the first frame of the movie is shown here to save space.)

```
In[98]:= movieframes = Table[Show[{cirptslines[[i]], cir, sinecurve},
           DisplayFunction -> $DisplayFunction], {i, 1, Length[cirptslines]}];
```



As described earlier, the frames of a movie can be shown together.

```
In[99]:= arraydisplay = Partition[movieframes, 4];
         Show[GraphicsArray[arraydisplay]];
```



## ■ Fours Types of Brackets in *Mathematica*

Four types of brackets have been described in this chapter: ( ...), { ...}, [ ...] and [[ ...]].

Round brackets ( ...) are used for mathematical grouping of terms.

```
In[101]:= 7 (3 + 4)
Out[101]= 49
```

Curly brackets { ...} are used for creating lists.

```
In[102]:= mylist = {π, e, 1}
Out[102]= {π, e, 1}
```

Square brackets [...] are used in defining functions and evaluating built-in, add-on and defined functions.

```
In[103]:= g[x_] = x + Sin[x];
          g[π / 2]
Out[104]= 1 + π/2
```

In[105]:= `Cos[6 π / 5]`

Out[105]= $\frac{1}{4}\left(-1-\sqrt{5}\right)$

Double square brackets [[...]] are used to extract parts of a list.

In[106]:= `mylist[[2]]`

Out[106]= e

**Do not attempt to interchange these symbols!** For example, using round brackets instead of square brackets when defining or evaluating a function will produce an undesired result.

In[107]:= `g (π)`

Out[107]= g π

## ■ Common Problems and How to Fix Them

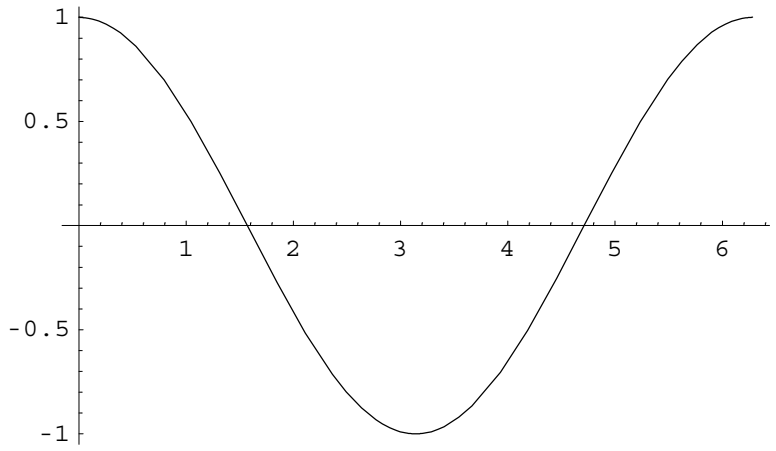
### Misinterpreting Warning Messages

If *Mathematica* thinks that you have executed a command containing an error, it will display a warning message.

```
In[108]:= Plot[Cos[x], {x, 0, 2 π];
          Syntax::bktmcp : Expression "{x, 0, 2 π" has no closing "}".
          Plot[Cos[x], {x, 0, 2 π];
```

In this case, a right curly bracket was forgotten and can be easily fixed.

```
In[108]:= Plot[Cos[x], {x, 0, 2 π}];
```




In the following example, study the warning message. In this case, there is absolutely nothing wrong with defining a list the name list, but *Mathematica* notices that this name is nearly the same as a *Mathematica* command called List with a capital L. In this case, you can ignore the warning message and move on. **Therefore, not all warning messages should be interpreted as error messages.**

```
In[109]:= list = {1, 2, 4}

          General::spell1 : Possible spelling error: new
             symbol name "list" is similar to existing symbol "List".

Out[109]= {1, 2, 4}
```

### Using an Add-On Function Before Loading it into Memory

Problems will occur if you attempt to use an add-on command before it has been loaded into memory.

For example, the add-on command `DayOfWeek[{year, month, day}]` will return the day of the week on which the given date occurred.

```
In[110]:= DayOfWeek[{1962, 6, 6}]

Out[110]= DayOfWeek[{1962, 6, 6}]
```

The command did not work because it was not loaded into memory first. The command is contained in a package called `Calender` inside a folder named `Miscellaneous`. Now the package is loaded and a warning message occurs.

```
In[111]:= << Miscellaneous`Calendar`

          DayOfWeek::shdw : Symbol DayOfWeek appears in multiple contexts
             {Miscellaneous`Calendar`, Global`}; definitions in context
             Miscellaneous`Calendar` may shadow or be shadowed by other definitions.
```

Suppose you ignore the warning message and you try to execute `DayOfWeek` again. Nothing happens!

```
In[112]:= DayOfWeek[{1962, 6, 6}]

Out[112]= DayOfWeek[{1962, 6, 6}]
```

In order to get the `DayOfWeek` command defined in the package to work correctly, you must first use a command called `Remove`.

```
In[113]:= Remove[DayOfWeek]
```

Now the command will work!

```
In[114]:= DayOfWeek[{1962, 6, 6}]

Out[114]= Wednesday
```

### Using Reserved *Mathematica* Words

As you already know, all *Mathematica* commands and constants begin with a capital letter. These commands are *reserved words* that cannot be used for anything else other than their intended purpose. So when assigning names or creating functions, it is generally a good idea to use names beginning with a lower case letter to avoid reserved *Mathematica* words. In the following example, if you try to assign the name `N` to the sum of one plus one, an error message will appear on the screen.

```
In[115]:= N = 1 + 1

          Set::wrsym : Symbol N is Protected.

Out[115]= 2
```

If N was not a reserved word, then the output of the following input statement would be 2.

```
In[116]:= N

Out[116]= N
```

If you wanted a quick reminder of the N command (introduced earlier), execute ?N.

```
In[117]:= ? N

      N[expr] gives the numerical value of expr. N[
        expr, n] attempts to give a result with n-digit precision.
```

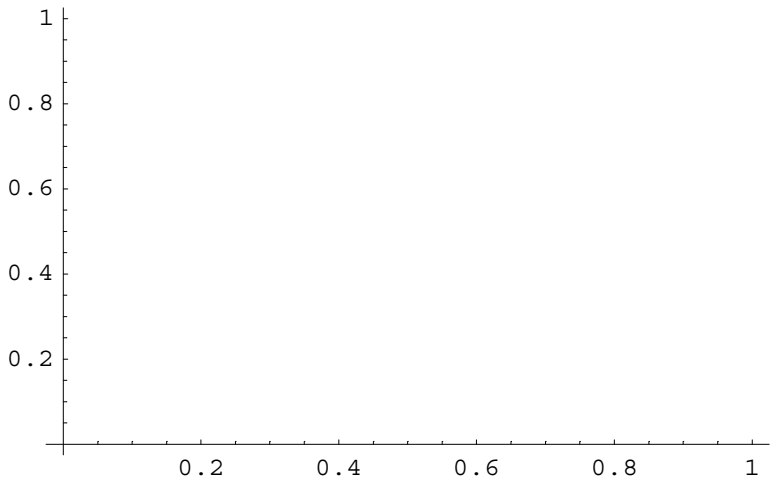
One way to resolve the problem would be to use a lower case letter.

```
In[118]:= n = 1 + 1;
          n

Out[119]= 2
```

### Failing to capitalize Reserved *Mathematica* commands

Another common problem when using *Mathematica* is failing to capitalize the first letter in a *Mathematica* command. In the following input line, sin[x] should be Sin[x] and therefore undesired results follow.

```
In[120]:= Plot[sin[x], {x, 0, 2 π}]

General::spell1 : Possible spelling error:
   new symbol name "sin" is similar to existing symbol "Sin".

Plot::plnr :
 sin[x] is not a machine-size real number at x = 2.617993877991494`*^-7.

Plot::plnr :
 sin[x] is not a machine-size real number at x = 0.25488992540742256`.

Plot::plnr :
 sin[x] is not a machine-size real number at x = 0.5328694051959509`.

General::stop :
 Further output of Plot::plnr will be suppressed during this calculation.
```




```
Out[120]= - Graphics -
```

In the following input cell, the first letter is capitalized, but since the `FindRoot` command consists of two words (i.e., `Find` and `Root`), the letter `r` should instead be `R`.

```
In[121]:= Findroot[x^2 - 11, {x, 2}]

General::spell1 : Possible spelling error: new symbol
   name "Findroot" is similar to existing symbol "FindRoot".

Out[121]= Findroot[-11 + x^2, {x, 2}]
```

Making the necessary correction leads to the desirable result.

```
In[122]:= FindRoot[x^2 - 11, {x, 2}]

Out[122]= {x → 3.31662}
```

This concludes your brief introduction to *Mathematica*. As you proceed through the following chapters, you may want to refer back to this chapter on occasion.

# CAS Exercise Examples for Chapter 1: Limits and Continuity

## ■ Section 1.1 Rates of Change and Limits

### Graphical Estimates of Limits

In Exercises 47 - 50, you are asked to estimate the value of $\lim_{x \to x_0} f(x)$ by plotting the graph of $y = f(x)$ near $x = x_0$ and to then evaluate the limit symbolically. The *Mathematica* command

```
Plot[f[x],{x,a,b}]
```

will plot the graph of the function $f(x)$ on the interval $[a, b]$. Adding the `PlotRange` option to form the command

```
Plot[f[x],{x,a,b},PlotRange->{c,d}]
```

will plot the graph of the function $f(x)$ with the range of $y$ values limited to the interval $[c, d]$. To compute $\lim_{x \to x_0} f(x)$ symbolically with *Mathematica*, use the command

```
Limit [f[x], x -> x0].
```

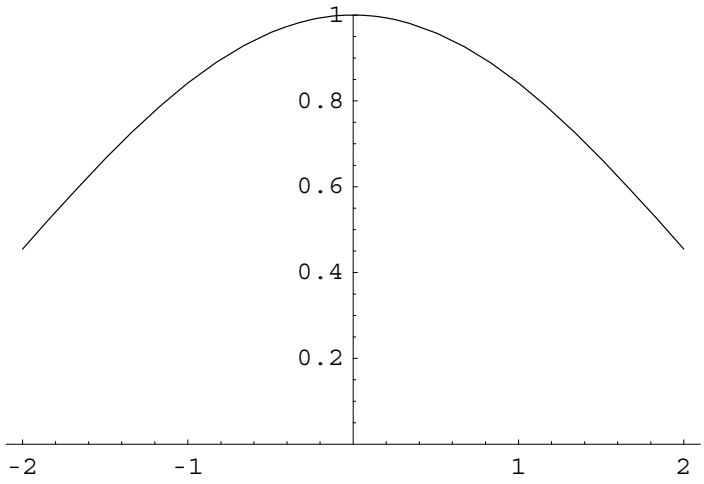
**Example:** Estimate $\lim_{x \to 0} \frac{\sin x}{x}$ by completing the following steps:

(a) Plot the function near the point $x_0$ being approached and from your plot, guess the value of the limit.

(b) Evaluate the limit symbolically.

**Part (a)** First we plot the function and from the graph you can estimate $\lim_{x \to 0} \frac{\sin x}{x} = 1$.

```
In[1]:= Plot[Sin[x]/x, {x, -2, 2}, PlotRange -> {0, 1}];
```




**Part (b)** To verify your guess in part (a), you can evaluate the limit with *Mathematica*.

```
In[2]:= Limit[Sin[x]/x, x → 0]
Out[2]= 1
```

### Finding Deltas Graphically

Executing the *Mathematica* command

```
Plot[f[x],{x,a,b},PlotRange->{{c,d},{e,f}}]
```

will plot the function on the interval $[a, b]$ and will then display the graph on an portion of the xy-plane with values of $x$ ranging from $x = c$ to $x = d$ and values of $y$ ranging from $y = e$ to $y = f$. Adding the option `PlotStyle->RGBColor[0,0,1]` will plot the curve in the color blue.

To plot several functions $f_1$, $f_2$, ..., $f_n$ simultaneously, execute the command

```
Plot[{f1[x], f2[x], ..., fn[x]},{x,a,b}]
```

and add the option

```
PlotStyle->{RGBColor[0,0,1],RGBColor[0,0,1],...,RGBColor[0,0,1]}
```

in order to plot each graph in the color blue.

Another *Mathematica* function used here is the command

```
ListPlot[{{x1, y1}, {x1, y1}, ..., {xn, yn}}, PlotJoined -> True]
```

which will connect the points $(x_1, y_1)$, $(x_2, y_2)$, ..., $(x_n, y_n)$. Adding the option `PlotStyle->RGBColor[0,0,1]` will plot the points in the color blue and/or adding the option `DisplayFunction->Identity` will suppress graph from being displayed on the screen.

Finally, the `Show` command is used to plot several graphs simultaneously.

**Example:** Use *Mathematica* to perform the following steps:

(a) Plot the function $y = \frac{x^5-32}{x-2}$ near the point $x_0 = 2$ being approached and guess the value of $\lim_{x\to x_0} f(x)$. Then verify your result with *Mathematica*.

(b) Using the value $\epsilon = 0.2$, graph the bounding lines $y_1 = L - \epsilon$ and $y_2 = L + \epsilon$ together with the function $y = \frac{x^5-32}{x-2}$ near $x_0$.

(c) From your graph in part (b), estimate $\delta > 0$ such that for all $x$

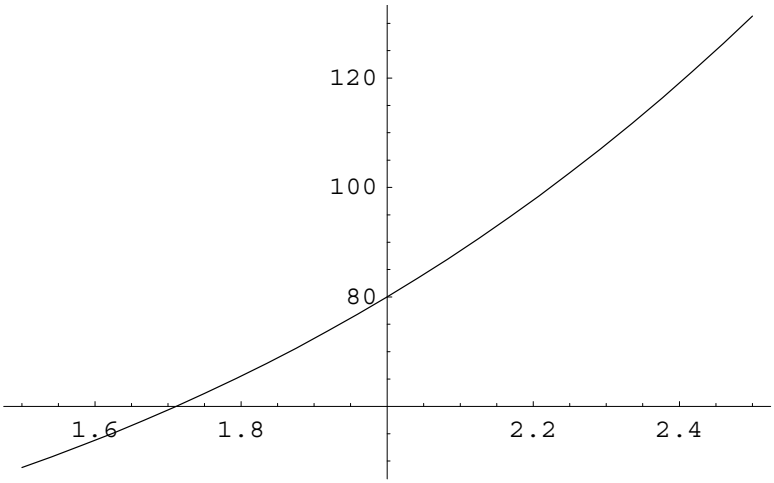
$$0 < |x - x_0| < \delta \Rightarrow |f(x) - L| < \epsilon.$$

Test you estimate by plotting $f$, $y_1$ and $y_2$ over the interval $0 < |x - x_0| < \delta$. For your viewing window, use

$x_0 - 2\delta \le x \le x_0 + 2\delta$ and $L - 2\epsilon \le y \le L + 2\epsilon$.

**Part (a)** In the following input cell, the current definition of $f(x)$ is cleared, then redefined and plotted on [1.5, 2.5]. It appears that the limit is 80.

```
In[3]:= Clear[f];
        f[x_] := (x^5 - 32)/(x - 2);
        Plot[f[x], {x, 1.5, 2.5}];
```

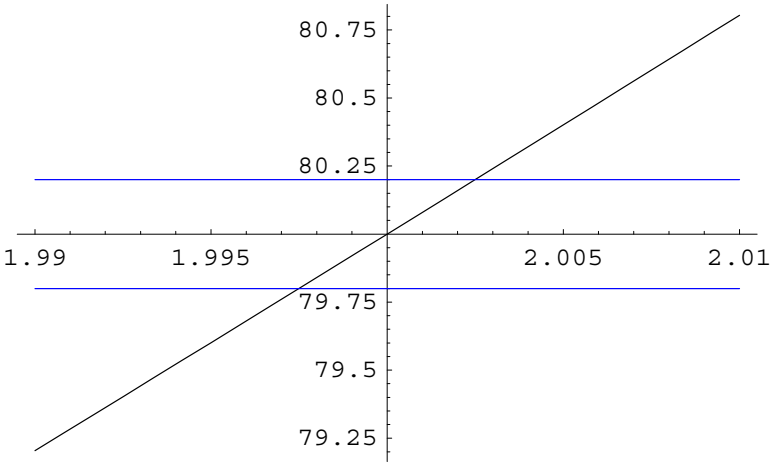



The `Limit` command is used to verify the limit symbolically.

```
In[4]:= Limit[f[x], x → 2]
Out[4]= 80
```

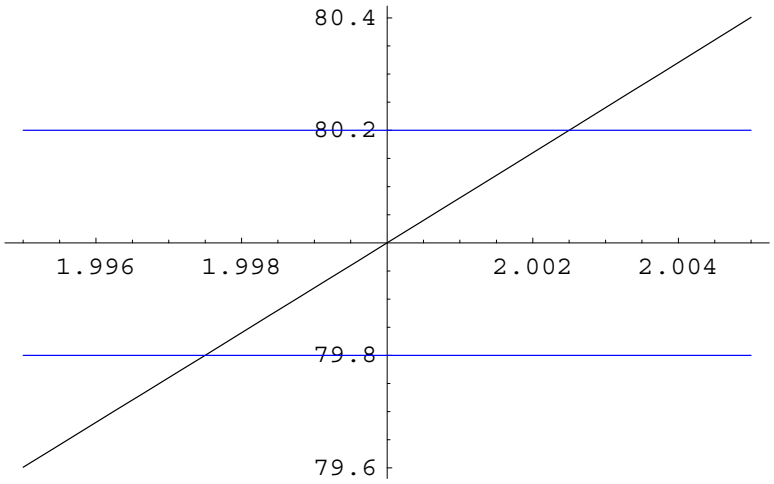
**Part (b)** In the following input cell, the value of $\epsilon$ is set equal to 0.2 and then the functions $f$, $y_1 = 80 - \epsilon$ and $y_2 = 80 + \epsilon$ are plotted together.

```
In[5]:= ϵ := 0.2;
        y1 := 80 - ϵ;
        y2 := 80 + ϵ;
        Plot[{f[x], y1, y2}, {x, 1.99, 2.01},
          PlotStyle -> {RGBColor[0, 0, 0], RGBColor[0, 0, 1], RGBColor[0, 0, 1]}];
```




**Part (c)** A new graph is now plotted over a smaller domain in order to estimate the value of $\delta$.

```
In[6]:= gr1 = Plot[{f[x], y1, y2}, {x, 1.995, 2.005},
          PlotStyle -> {RGBColor[0, 0, 0], RGBColor[0, 0, 1], RGBColor[0, 0, 1]}];
```




From the graph, you can see that if we choose $\delta = 0.002$, then it appears that $|f(x) - 80| < \epsilon$ whenever $|x - 2| < \delta$.

In the following input cell, a box is defined in which the function $f$ should pass through without ever crossing the top or the bottom of the box. In each of the assignment statements, a delayed equals (:=) is used so that x1, x2 and $\delta\epsilon$box are not are not assigned values until you call them later in the session.

```
In[7]:= x1 := 2 - δ;
        x2 := 2 + δ;
        δϵbox :=
          ListPlot[{{x1, y1}, {x1, y2}, {x2, y2}, {x2, y1}, {x1, y1}}, PlotJoined -> True,
           PlotStyle -> RGBColor[0, 0, 1], DisplayFunction -> Identity];
```

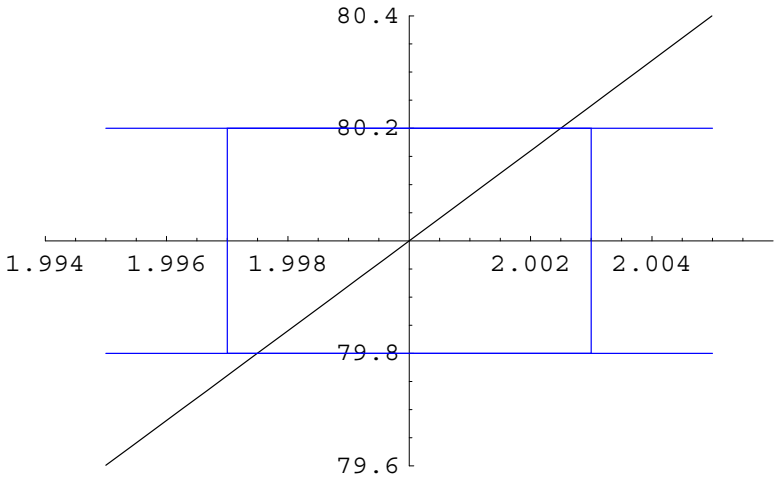
The names gr1 and $\delta\epsilon$box were assigned to the previous two graphs, so the command

`Show[{gr1,δϵbox}]`

can be used to show the two graphs simultaneously.

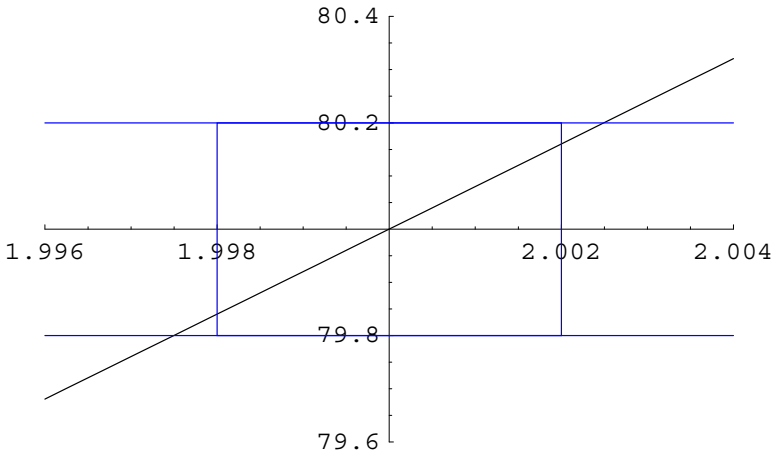
Suppose instead that you had selected $\delta$ to be 0.003. From the graph below, you can see that the function lies outside the interval $[80 - \epsilon, 80 + \epsilon]$.

```
In[8]:= δ = .003;
        Show[{gr1, δεbox}, PlotRange -> {{2 - 2 δ, 2 + 2 δ}, {80 - 2 ε, 80 + 2 ε}}];
```




In fact, $\delta$ should be smaller as illustrated below.

```
In[9]:= Clear[δ];
        δ = .002;
        Show[{gr1, δεbox}, PlotRange -> {{2 - 2 δ, 2 + 2 δ}, {80 - 2 ε, 80 + 2 ε}}];
```




So if $\delta = 0.002$, then $0 < |x - x_0| < \delta \Rightarrow |f(x) - 80| < \epsilon$.

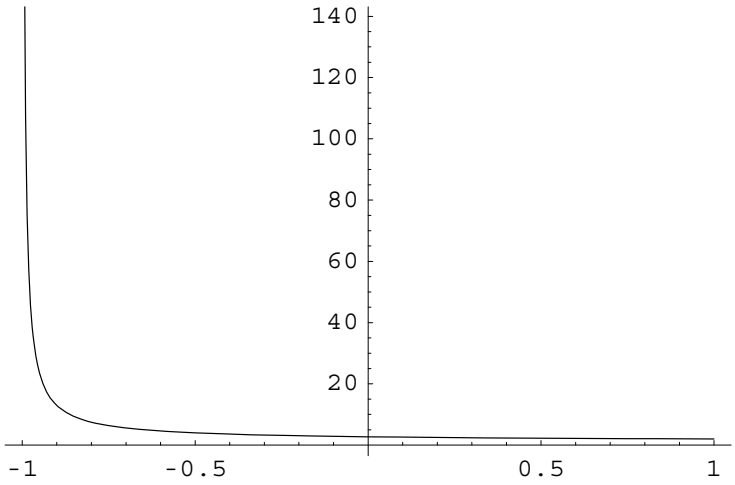
## ■ Section 1.2 Finding Limits and One-Sided Limits

In Exercises 43 & 44, you are asked to estimate limits by zooming in on the graph of a function. A similar example follows here.

**Example:** Graph the function $f(x) = (1 + x)^{1/x}$ to estimate $\lim_{x \to 0} f(x)$, zooming in on the origin as necessary.

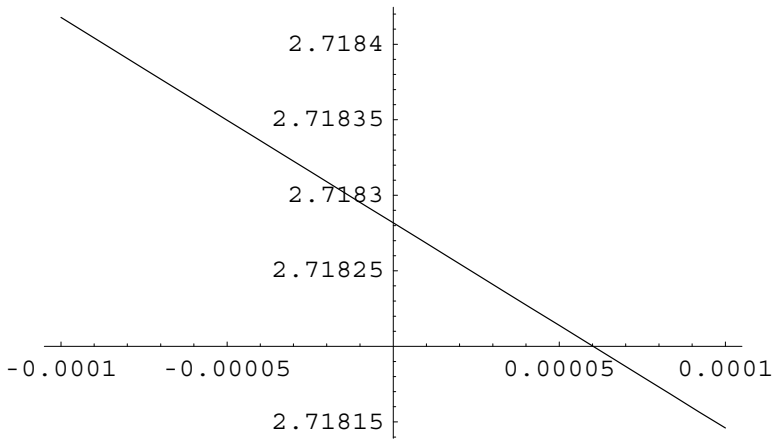
The function is defined and plotted below.

```
In[10]:= Clear[f];
         f[x_] := (1 + x)^(1/x);
         Plot[f[x], {x, -1, 1}];
```




Choosing a smaller interval in the `Plot` command will lead to a better estimate of the limit. From the following graph, you can see that $\lim_{x\to 0} f(x) \approx 2.71828$.

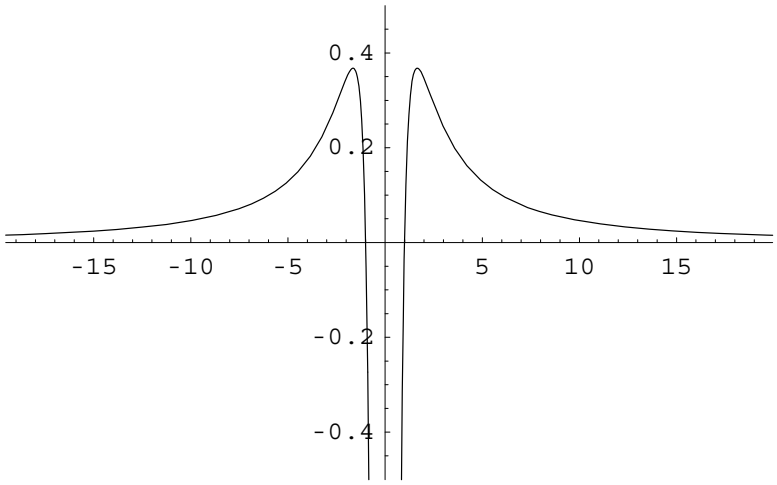
```
In[11]:= Plot[f[x], {x, -.0001, .0001}];
```




## ■ Section 1.3 Limits involving Infinity

In Exercises 47 - 50, you are asked to graph a given curve and describe what you see. Consider the graph of the function $f(x) = \frac{\ln(x^2)}{x^2}$ plotted below. Note that `Log[x]` is the *Mathematica* command for the function $\ln(x)$.

In[12]:=
```
Clear[f];
f[x_] := Log[x^2]/x^2;
Plot[f[x], {x, -30, 30}, PlotRange → {-.5, .5}];
```




From the graph it appears that $\lim_{x\to\infty} f(x) = 0$, $\lim_{x\to-\infty} = 0$, $\lim_{x\to 0} f(x) = -\infty$.

# CAS Exercise Examples for Chapter 2: Derivatives

## ■ Section 2.1 The Derivative of a Function

The *Mathematica* command `Simplify[expression]` will simplify an algebraic expression.

**Example:** For the function $f(x) = x^3 - 3x^2 - 5$, do the following:

(a) Define the difference quotient $q$ at a general point $x$, with general stepsize $h$.

(b) Take the limit of the difference quotient as $h \to 0$ and let $m(x)$ represent the function obtained from computing the limit.

(c) On the same graph, plot $m(x)$ together with $q$ for values of $h = 1,\ 0.5,$ and $0.25$.

(d) Plot $y = f(x)$ together with the graph of the tangent line to $f$ at $x_0 = 3$.

(e) Substitute various values for $x$ larger and smaller than $x_0$ into $m(x)$.

**Part (a)** Begin by defining $f$ and $q$.

```
In[1]:= f[x_] = x^3 - 3 x^2 - 4;
        q = Simplify[(f[x + h] - f[x])/h]
Out[1]= h^2 + 3 h (-1 + x) + 3 (-2 + x) x
```

**Part (b)** From the simplified form of $q$ found in part (a), can you determine of $q$ as $h \to 0$?

Your result can be verified using the `Limit` command.

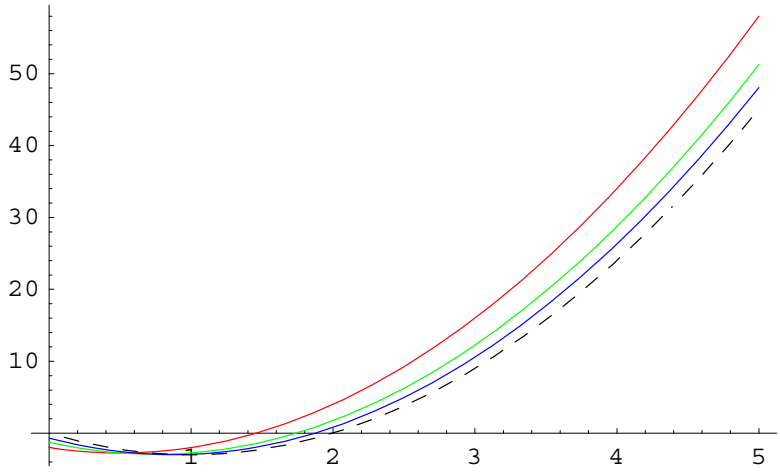
```
In[2]:= m[x_] = Limit[q, h → 0]
Out[2]= 3 (-2 + x) x
```

**Part (c)** The difference quotient is plotted below for the specified values of $h$ and then $m(x)$ is plotted.

The `ReplaceAll` command (`/.`) is used to replace $q$ with specific values of $h$. For example, `q/.h->1` will output the value of $q$ when $h$ is 1. Since the option `DisplayFunction→Identity` is used to suppress all the individual graphs from being displayed, then `DisplayFunction→$DisplayFunction` must be an included option in the `Show` command to display all the functions on the same graph. The option `DisplayFunction→$DisplayFunction` instructs *Mathematica* to display a previously suppressed graph. Also note that RGBColor[1,0,0],RGBColor[0,1,0] , and RGBColor[0,0,1] represent the colors red, green and blue, respectively.

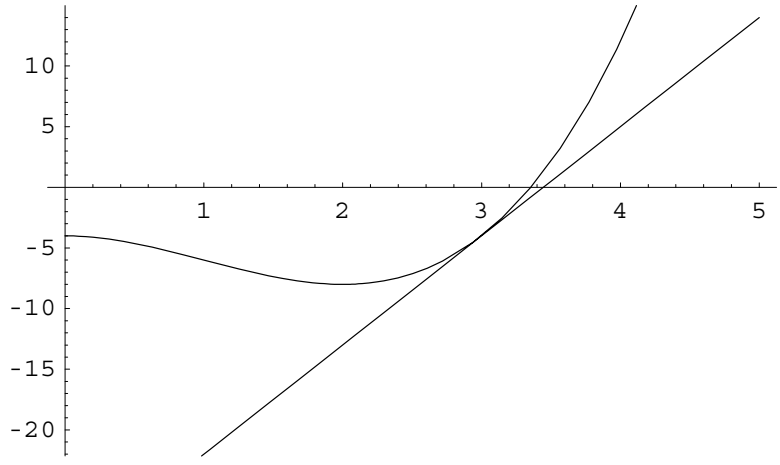
Study the following input and output. What is happening to the graph of $q$ as $h \to 0$? Why?

```
In[3]:= p1 = Plot[q /. h -> 1, {x, 0, 5},
          PlotStyle -> RGBColor[1, 0, 0], DisplayFunction → Identity];
        p2 = Plot[q /. h -> .5, {x, 0, 5}, PlotStyle -> RGBColor[0, 1, 0],
          DisplayFunction → Identity];
        p3 = Plot[q /. h -> .25, {x, 0, 5}, PlotStyle -> RGBColor[0, 0, 1],
          DisplayFunction → Identity];
        p4 = Plot[m[x], {x, 0, 5}, PlotStyle -> Dashing[{.02}],
          DisplayFunction -> Identity];
        Show[{p1, p2, p3, p4}, DisplayFunction -> $DisplayFunction];
```




**Part (d)** Since $m(x_0)$ represents the slope of the line tangent to $y = f(x)$ at $(x_0, y_0)$, the equation of the tangent line at $(3, f(3))$ is easily found and plotted together with $y = f(x)$ in the following cell.

```
In[4]:= y = m[3] (x - 3) + f[3];
        Plot[{f[x], y}, {x, 0, 5}];
```




**Part (e)** The command `Table[m[x],{x,1,5,.2}]` will form a list `{m[1], m[1.2], m[1.4], ..., m[5]}`.

Compare the resulting numbers with the graph of $y = f(x)$. Do these numbers make sense when compared with the graph of $y = f(x)$?

```
In[5]:= Table[m[x], {x, 1, 5, .2}]
Out[5]= {-3, -2.88, -2.52, -1.92, -1.08, 0., 1.32, 2.88, 4.68, 6.72, 9.,
  11.52, 14.28, 17.28, 20.52, 24., 27.72, 31.68, 35.88, 40.32, 45.}
```

### Computing and Graphing Derivatives with *Mathematica*

There are several ways of computing $f'(x)$ with *Mathematica*. Two ways are illustrated below, after first defining a new function.

```
In[6]:= Clear[f];
        f[x_] = 5 x^3 - 15 x - 2;
        f'[x]
Out[6]= -15 + 15 x^2
```

Here is another way of computing $f'(x)$.

```
In[7]:= D[f[x], x]
Out[7]= -15 + 15 x^2
```

Computing higher order derivatives such as $f''(x)$, $f'''(x)$ and so forth are just as easy. The command `D[f[x],{x,n}]` will compute the nth derivative of $f(x)$. Study the following example.

```
In[8]:= f''[x]
Out[8]= 30 x

In[9]:= D[f[x], {x, 3}]
Out[9]= 30
```

## ■ Section 2.2 The Derivative as a Rate of Change

### Simulation of Motion on a Vertical Line

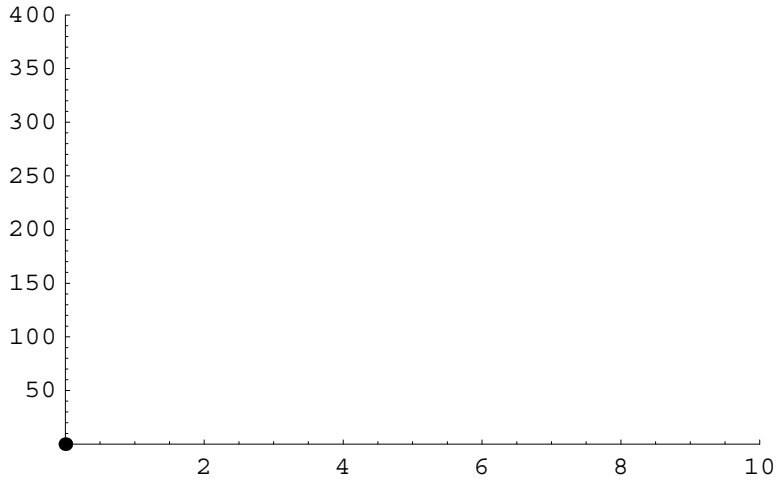
Consider the motion of the rock in Example 5 in your textbook. Suppose we wish to simulate the motion of the rock. Plotting the points $(0, y(t))$ one at a time, where $y(t) = 160t - 16t^2$ for values of $t = 0, .4, .8, \ldots, 10.0$ and then animating the graphs will produce a simulation of the motion of the rock.

After defining $x(t)$ and $y(t)$, a table of ordered pairs $(0, y(t))$, which is represented with *Mathematica* by `{0,y[t]}`, is created for the position of the rock above the ground at time $t$.

```
In[10]:= Clear[x, y, t];
         x[t_] = t; y[t_] = 160 t - 16 t^2;
         rock = Table[{0, y[t]}, {t, 0, 10, .4}]
Out[10]= {{0, 0}, {0, 61.44}, {0, 117.76}, {0, 168.96}, {0, 215.04}, {0, 256.}, {0, 291.84},
  {0, 322.56}, {0, 348.16}, {0, 368.64}, {0, 384.}, {0, 394.24}, {0, 399.36},
  {0, 399.36}, {0, 394.24}, {0, 384.}, {0, 368.64}, {0, 348.16}, {0, 322.56},
  {0, 291.84}, {0, 256.}, {0, 215.04}, {0, 168.96}, {0, 117.76}, {0, 61.44}, {0, 0.}}
```

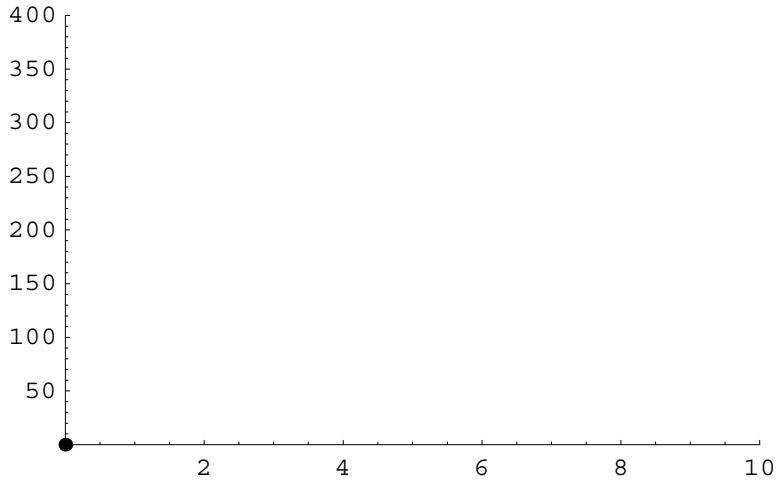
The `ListPlot` command is now used to plot the points one at a time. The output (of which only one frame is shown in the hard copy of this manual), can be animated by double-clicking on any one of the frames. The option `PlotStyle->PointSize[.02]` is used to increase the displayed size of each point. Also note that `rock[[i]]` represents the $i^{th}$ ordered pair in the list `rock`.

```
In[11]:= rockmotion = Table[ListPlot[{rock[[i]]}, PlotStyle -> PointSize[.02],
           PlotRange -> {{0, 10}, {0, 400}}], {i, 1, Length[rock]}];
```




In the following, a set of ordered pairs $(x(t),\ y(t))$ where $x(t) = t$ and $y(t) = 160\,t - 16\,t^2$ is plotted on point at a time together with (0,y(t)). Again, only the first frame is displayed below in the hard copy of this manual. The frames can be animated to show how the height of the rock corresponds to the point of the graph of $s = 160\,t - 16\,t^2$.

```
In[12]:= graphpoints = Table[{x[t], y[t]}, {t, 0, 10, .4}];
         rockmotion =
           Table[ListPlot[{rock[[i]], graphpoints[[i]]}, PlotStyle -> PointSize[.02],
             PlotRange -> {{0, 10}, {0, 400}}], {i, 1, Length[rock]}];
```



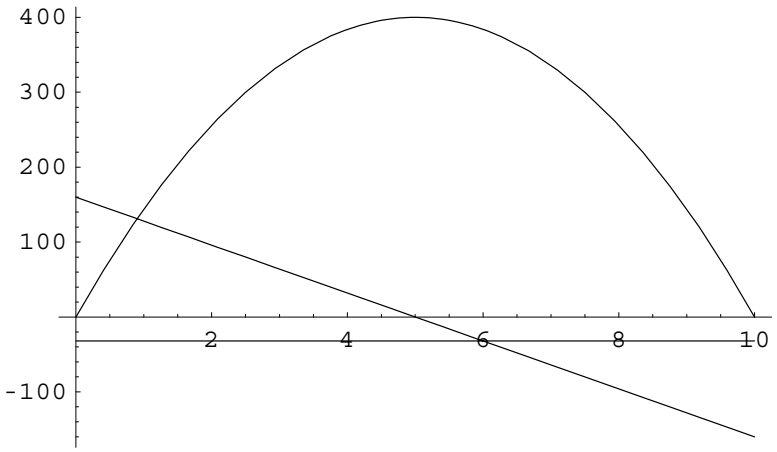

### Graphing the position function $s(t)$, the velocity function $v(t)$ and the acceleration function $a(t)$.

Here is an example of how to use *Mathematica* to plot a position vector with its velocity and acceleration functions.

**Example:** For the position function $s = 160\,t - 16\,t^2$, plot its graph together with the velocity and acceleration functions.

The solution is a simple as defining the position function and then plotting it with its first and second derivatives. What information does the graph reveal?

```
In[13]:= s[t_] = 160 t - 16 t^2;
         Plot[{s[t], s'[t], s''[t]}, {t, 0, 10}];
```

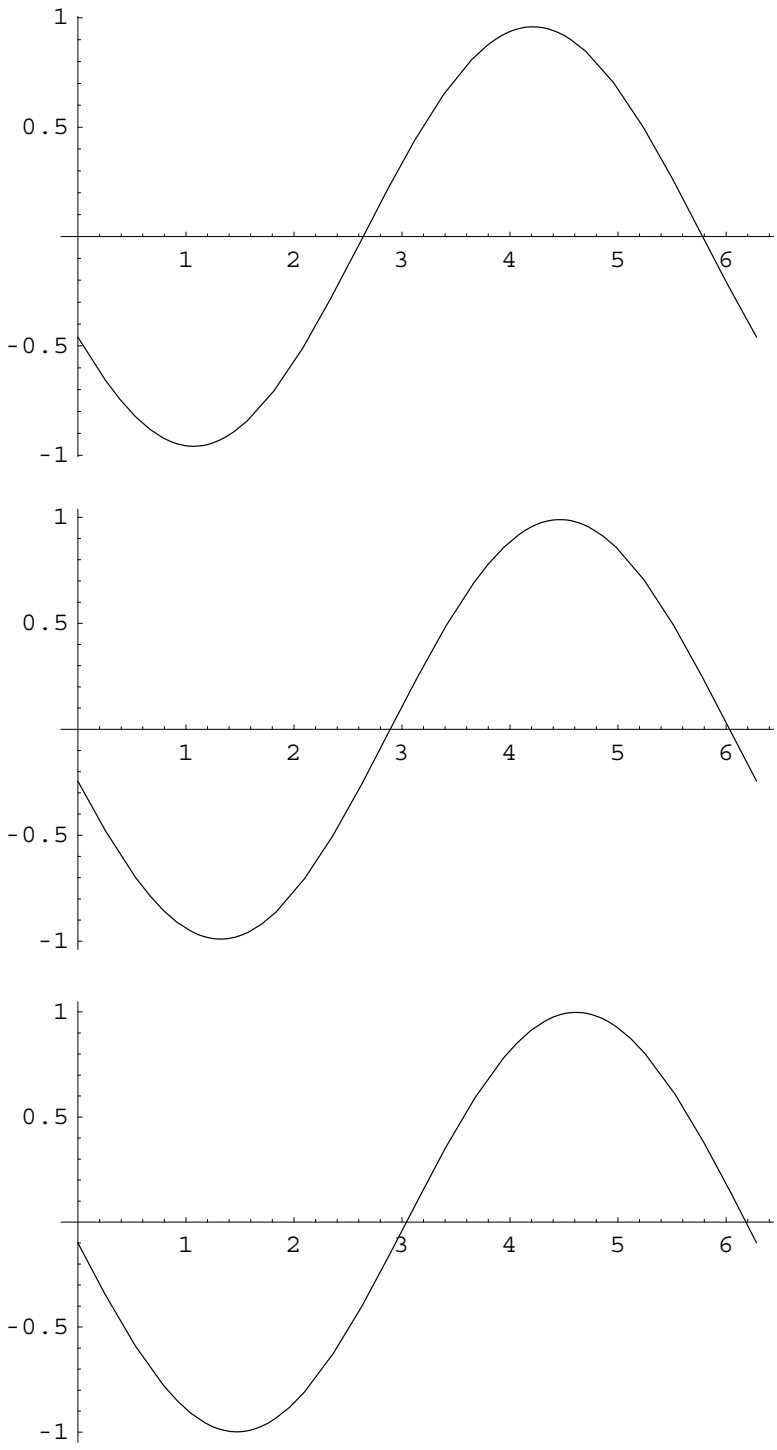



## ■ Section 2.5 The Chain Rule

### The Derivative of Cos(*x*)

Recall that $\frac{d}{dx}\cos(x) = \lim_{h\to 0}\frac{\cos(x+h)-\cos(x)}{h}$. To verify that $\frac{d}{dx}\cos(x) = -\sin(x)$, the difference quotient $\frac{\cos(x+h)-\cos(x)}{h}$ is plotted below for $h = 1$, .5 and .2. Examine the output of the following plot commands and explain what appears to be happening as $h \to 0$. I.e., do the graphs appear to be converging to a familiar function?

```
In[14]:= Clear[q];
         q = (Cos[x + h] - Cos[x]) / h;
         p1 = Plot[q /. h → 1., {x, 0, 2 π}];
         p2 = Plot[q /. h → .5, {x, 0, 2 π}];
         p3 = Plot[q /. h → .2, {x, 0, 2 π}];
```


1
0.5
-0.5
-1
1
2
3
4
5
6
1
0.5
-0.5
-1
1
2
3
4
5
6
1
0.5
-0.5
-1
1
2
3
4
5
6

### Parametrized Curves

The following example is similar to Exercises 75 - 80 in your text.

**Example:** Consider the parametrized curve whose equations are $x(t) = t + \cos(t)$ and $y(t) = t - \sin(t)$ where $0 \le t \le 6\pi$.

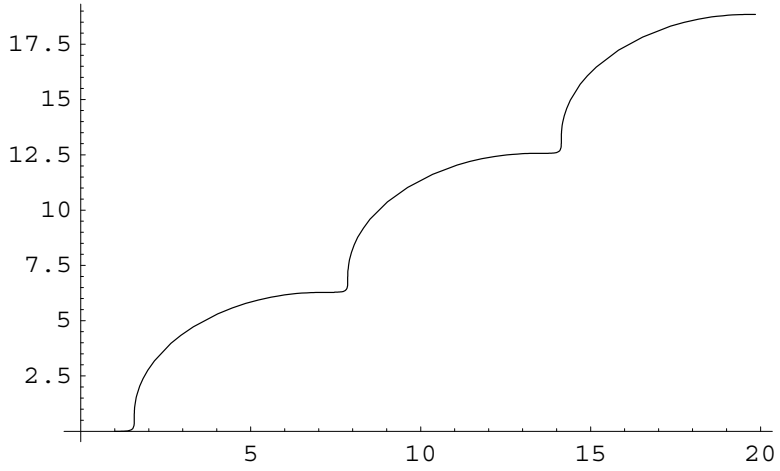
(a) Plot the curve over the given interval.

(b) Find $\frac{dy}{dx}$ and $\frac{d^2y}{dx^2}$.

(c) Find an equation for the tangent line to the curve at the point where $t_0 = 4$.

**Part (a)** The parametric curves are defined and then plotted using the following commands.

```
In[15]:= Clear[x, y];
         x[t_] = t + Cos[t]; y[t_] = t - Sin[t];
         funplot = ParametricPlot[{x[t], y[t]}, {t, 0, 6 π}];
```




**Part (b)** Next, $\frac{dy}{dx}$ and $\frac{d^2y}{dx^2}$ are defined. Since $\frac{dy}{dx}$ is used in part (c) to find the slope of a tangent line, the derivative is defined to be `yprime[t]` in the following cell.

```
In[16]:= yprime[t_] = y'[t] / x'[t]
```

$$\text{Out[16]=}\ \frac{1 - \text{Cos}[t]}{1 - \text{Sin}[t]}$$

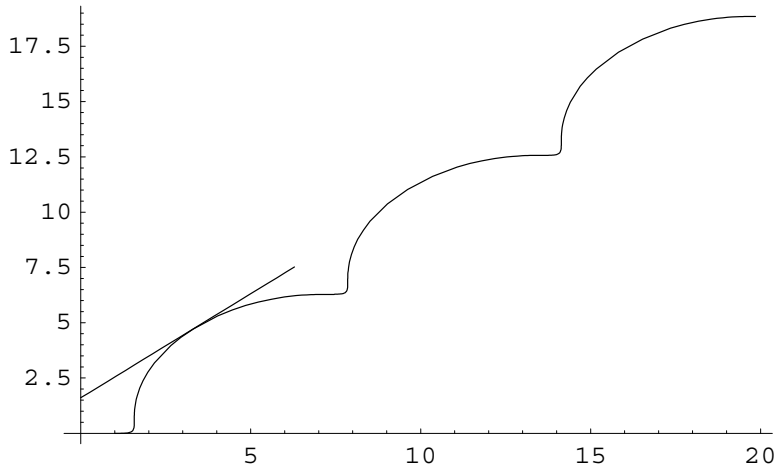
Next, the second derivative is defined and simplified.

```
In[17]:= Simplify[ yprime'[t] / x'[t] ]
```

$$\text{Out[17]=}\ \frac{2\,\text{Sin}\left[\frac{t}{2}\right]}{\left(\text{Cos}\left[\frac{t}{2}\right] - \text{Sin}\left[\frac{t}{2}\right]\right)^5}$$

**Part (c)** Now the tangent line is computed and then the `Show` command is used to show the graph of the parametrized curve and the tangent line simultaneously.

```
In[18]:= tanline = yprime[4.] (x - x[4.]) + y[4.];
         tanlineplot = Plot[tanline, {x, 0, 2 π}, DisplayFunction -> Identity];

In[19]:= Show[funplot, tanlineplot];
```




## ■ Section 2.6 Implicit Differentiation

The following example illustrates how to manipulate implicit equations with *Mathematica*.

**Example:** Given the equation $(x-1)^4 = x^2 - y^2$, use *Mathematica* to do the following.

(a) Solve the equation of $y$ and then differentiate the result to find $\frac{dy}{dx}$.

(b) Use implicit differentiation to find $\frac{dy}{dx}$ and verify that the result agrees the derivative found in part a.

(c) Plot the graph of the equation together with the line tangent to the curve at $\left(\frac{7}{4}, \frac{\sqrt{703}}{16}\right)$.

**Part (a)** Begin by assigning the equation the name `eq` and then use the `Solve` command to solve the equation for ***y*.**

```
In[20]:= Clear[x, y, eq];
         eq = (x - 1)^4 == x^2 - y^2;
         solutions = Solve[eq, y]
```

$$\text{Out[20]=}\ \left\{\left\{y \to -\sqrt{-1 + 4\,x - 5\,x^2 + 4\,x^3 - x^4}\right\},\ \left\{y \to \sqrt{-1 + 4\,x - 5\,x^2 + 4\,x^3 - x^4}\right\}\right\}$$

The solutions for $y$ can be extracted using the commands `solutions[[1,1,2]]` and `solutions[[2,1,2]]`. Next, the `D` command is used to find and simplify $\frac{dy}{dx}$ for each solution.

```
In[21]:= D[solutions[[1, 1, 2]], x] // Simplify
         D[solutions[[2, 1, 2]], x] // Simplify
```

$$\text{Out[21]=}\ \frac{-2 + 5\,x - 6\,x^2 + 2\,x^3}{\sqrt{-1 + 4\,x - 5\,x^2 + 4\,x^3 - x^4}}$$

$$\text{Out[22]=}\ \frac{2 - 5\,x + 6\,x^2 - 2\,x^3}{\sqrt{-1 + 4\,x - 5\,x^2 + 4\,x^3 - x^4}}$$

**Part (b)** The `Dt` command can be used to find $\frac{dy}{dx}$ using implicit differentiation. The result is an equation which is arbitrarily assigned the name `implicitder`.

```
In[23]:= implicitder = Dt[eq, x]
```

$$\text{Out[23]= } 4\,(-1+x)^3 == 2\,x - 2\,y\,\text{Dt}[y, x]$$

The *Mathematica* notation `Dt[y,x]` above represents $\frac{dy}{dx}$. The *Mathematica* command

`Solve[implicitder,Dt[y,x]]`

in the following input cell will solve the equation for `Dt[y,x]`.

```
In[24]:= impsolution = Solve[implicitder, Dt[y, x]]
```

$$\text{Out[24]= } \left\{\left\{\text{Dt}[y, x] \to -\frac{-2 + 5\,x - 6\,x^2 + 2\,x^3}{y}\right\}\right\}$$

To verify that the derivative obtained using implicit differentiation is the same as the derivative obtained earlier, replace the value of y with the explicit solutions found earlier.

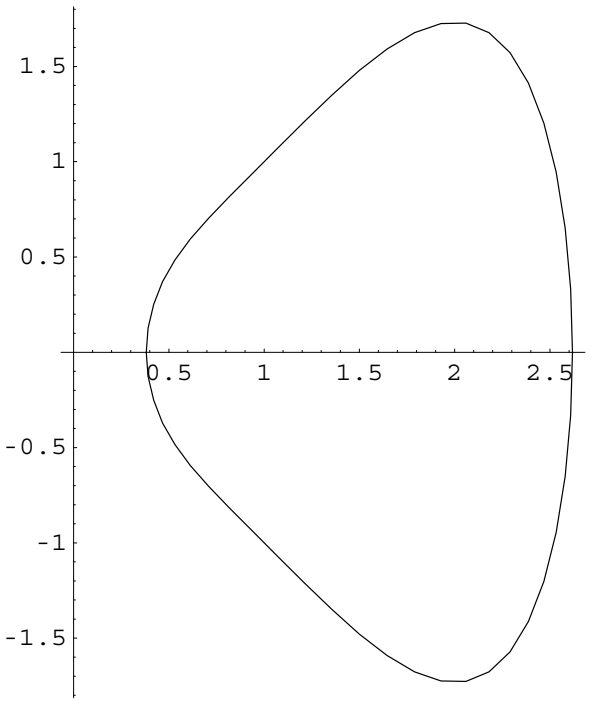
```
In[25]:= impsolution /. y → solutions[[1, 1, 2]]
         impsolution /. y → solutions[[2, 1, 2]]
```

$$\text{Out[25]= } \left\{\left\{-\frac{4 - 10\,x + 12\,x^2 - 4\,x^3}{2\sqrt{-1 + 4\,x - 5\,x^2 + 4\,x^3 - x^4}} \to \frac{-2 + 5\,x - 6\,x^2 + 2\,x^3}{\sqrt{-1 + 4\,x - 5\,x^2 + 4\,x^3 - x^4}}\right\}\right\}$$

$$\text{Out[26]= } \left\{\left\{\frac{4 - 10\,x + 12\,x^2 - 4\,x^3}{2\sqrt{-1 + 4\,x - 5\,x^2 + 4\,x^3 - x^4}} \to -\frac{-2 + 5\,x - 6\,x^2 + 2\,x^3}{\sqrt{-1 + 4\,x - 5\,x^2 + 4\,x^3 - x^4}}\right\}\right\}$$

**Part (c)** To plot the equation, the `ImplicitPlot` command found in the package called **ImplicitPlot**, must be loaded into memory first using the `Needs` command.

```
In[27]:= << Graphics`ImplicitPlot`
         eqgraph = ImplicitPlot[eq, {x, -4, 4}];
```



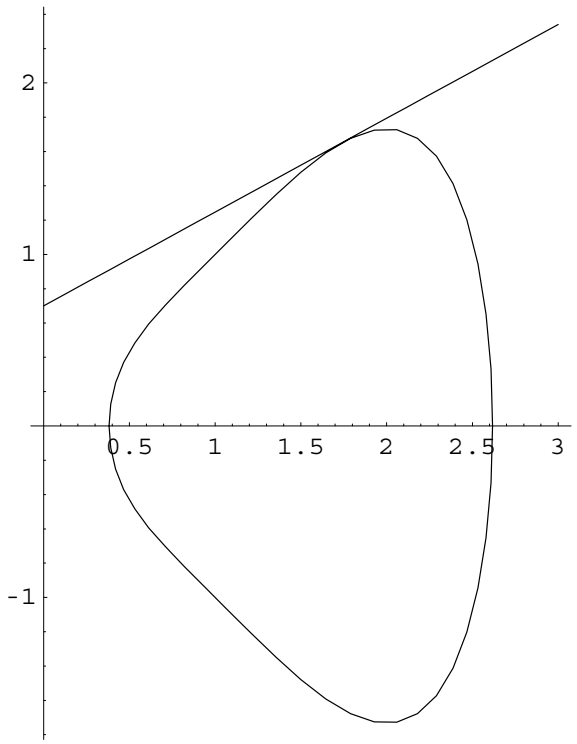

To find the equation of the line tangent to the equation at $\left(\frac{7}{4}, \frac{\sqrt{703}}{16}\right)$, $(x_0, y_0)$ is assigned the value $\left(\frac{7}{4}, \frac{\sqrt{703}}{16}\right)$ and then the implicit solution for $\frac{dy}{dx}$ is replaced with this ordered pair using the `ReplaceAll (/.)` command.

```
In[29]:= x0 = 7/4; y0 = Sqrt[703]/16;
         m = impsolution[[1, 1, 2]] /. {x → x0, y → y0}
```

$$\text{Out[29]=} \quad \frac{29}{2\sqrt{703}}$$

Now the tangent line is formed and the plotted. The `Show` command is then used to plot the implicit curve and the tangent line together on a single graph.

In[30]:=
```
Clear[x, y];
y = m (x - x0) + y0;
tangraph = Plot[y, {x, 0, 3}, DisplayFunction -> Identity];
Show[{eqgraph, tangraph}];
```



# CAS Exercise Examples for Chapter 3: Applications of Derivatives

---

## ■ Section 3.1 Extreme Values of Functions

### Finding Solutions of $g(x) = 0$

By graphing a function $g(x)$, the approximate locations of the roots of $g(x)$ can be found. Then the command `FindRoot[g[x], {x, x0}]` can be used to find a solution to $g(x) = 0$ where $x_0$ is the approximate location of the given root determined by observing the graph of $y = g(x)$. The `FindRoot` command will be useful in the following example.

### Using *Mathematica* to locate absolute extrema on a closed interval

**Example:** Given the function $f(x) = x^2 \sqrt[3]{x+1} - \sin(x)$ and the closed interval $[-1, 1]$, perform each of the following.

(a) Plot the function over the interval to see its general behavior there.

(b) Find interior points where $f' = 0$.

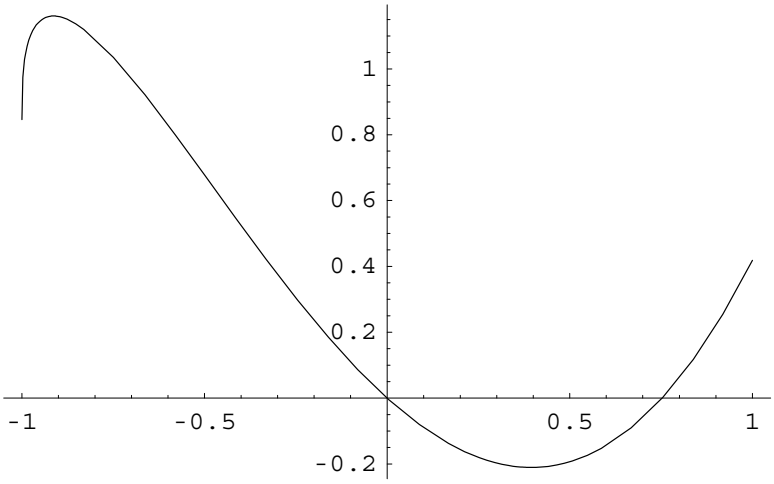
(c) Find the interior points where $f'$ doesn't exist.

(d) Evaluate the function at all points found in parts (b) and (c) and at the endpoints of the interval.

(e) Find the functions absolute extreme values on the interval and identify where they occur.

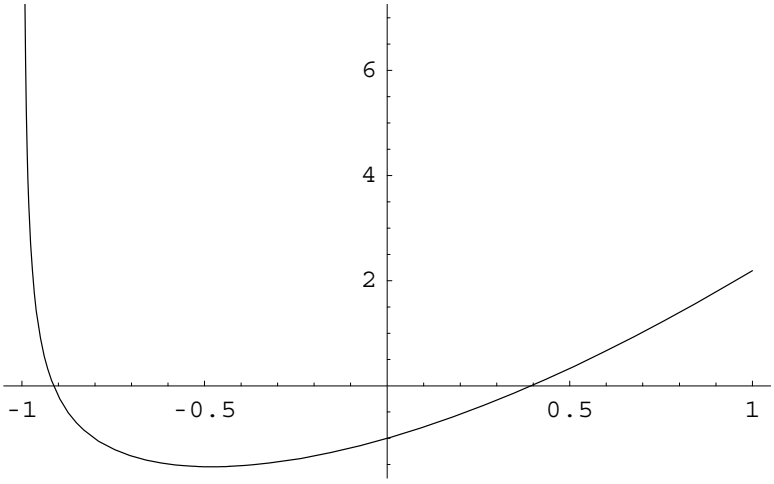
Part (a) The function is defined and plotted first.

```
In[1]:= f[x_] = x^2 (x + 1)^(1/3) - Sin[x];
        Plot[f[x], {x, -1, 1}];
```




To see the approximate locations where f '(x)=0, plot the derivative.

```
In[3]:= Plot[f'[x], {x, -1, 1}];
```




The derivative is zero at about $x = -0.95$ and $x = 0.4$. The following `FindRoot` command will locate the negative root.

```
In[4]:= sol1 = FindRoot[f'[x] == 0, {x, -.95}]
Out[4]= {x → -0.913313}
```

From the graph of $f\,'(x)$, it is clear that another root lies near $x = 0.4$.

```
In[5]:= sol2 = FindRoot[f'[x] == 0, {x, .4}]
Out[5]= {x → 0.394546}
```

**Part (c)** From observing the function $f'(x)$ in the following output cell, you can see that $f'(x)$ is never undefined at point in the interior of $[-1, 1]$.

```
In[6]:= Simplify[f'[x]]
Out[6]= x (6 + 7 x)/(3 (1 + x)^(2/3)) - Cos[x]
```

**Part (d)** Now we evaluate the function at the two values of $x$ where $f'(x) = 0$ and at the endpoints. Note that `N[f[x]]` or equivalently, `f[x]//N` will give a numerical approximation of the value of $f(x)$. Recall that the command `sol1[[1,2]]` will extract the first solution found in part (b).

```
In[7]:= f[sol1[[1, 2]]]
        f[sol2[[1, 2]]]
        f[-1] // N
        f[1] // N
Out[7]= 1.1607
Out[8]= -0.210473
Out[9]= 0.841471
Out[10]= 0.41845
```

**Part (e)** From observing the output in part (d), you can see that on the interval $[-1, 1]$, the function has an absolute maximum of approximately 1.1607 at $x \approx -0.913313$ and an absolute minimum value of about $-0.210473$ at $x \approx 0.394546$.

■

## Section 3.2 The Mean Value Theorem and Differential Equations

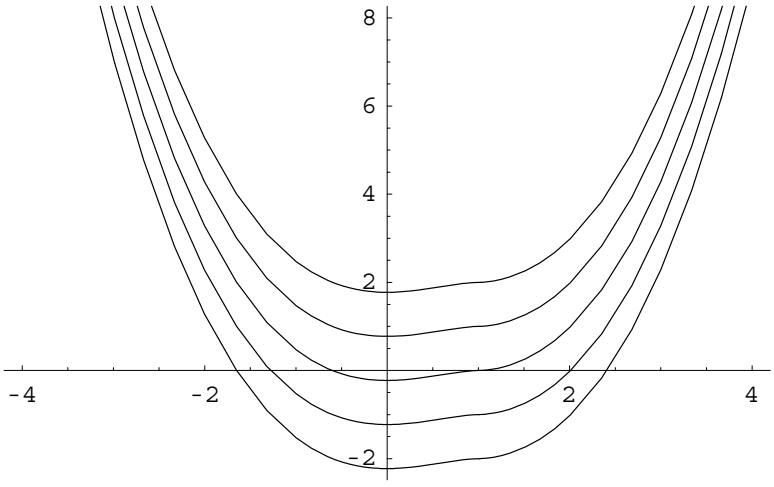
### Solving Differential Equations with *Mathematica*

The command `DSolve[eqn, y[x], x]` will solve a differential equation involving $y'(x)$ for $y$ in terms of $x$. For example, the following input command is used to solve the differential equation $y' = x\sqrt[3]{(x-1)^2}$ for $y$ in terms of $x$.

```
In[11]:= sol = DSolve[y'[x] == x (x - 1)^2^(1/3), y[x], x]
Out[11]= {{y[x] → ((-1 + x)^2)^(1/3) (-9/40 - 3 x/20 + 3 x^2/8) + C[1]}}
```

The solution is extracted using the command `sol[[1,1,2]]`. The value `C[1]` in the solution represents an arbitrary constant. In the following input cell, the command `/.` is used to replace `C[1]` with specific values and the solution is then plotted for these specific values of `C[1]`.

```
In[12]:= Plot[{sol[[1, 1, 2]] /. C[1] → -2,
    sol[[1, 1, 2]] /. C[1] → -1, sol[[1, 1, 2]] /. C[1] → 0,
    sol[[1, 1, 2]] /. C[1] → 1, sol[[1, 1, 2]] /. C[1] → 2}, {x, -4, 4}];
```




Now suppose you want to find and plot the solution passing through $(\frac{1}{2}, -2)$. Begin by letting y equal the solution found earlier with x replaced with $\frac{1}{2}$.

```
In[13]:= y = sol[[1, 1, 2]] /. x → 1/2
```

$$\text{Out[13]=} \quad -\frac{33}{160\ 2^{2/3}} + \texttt{C[1]}$$

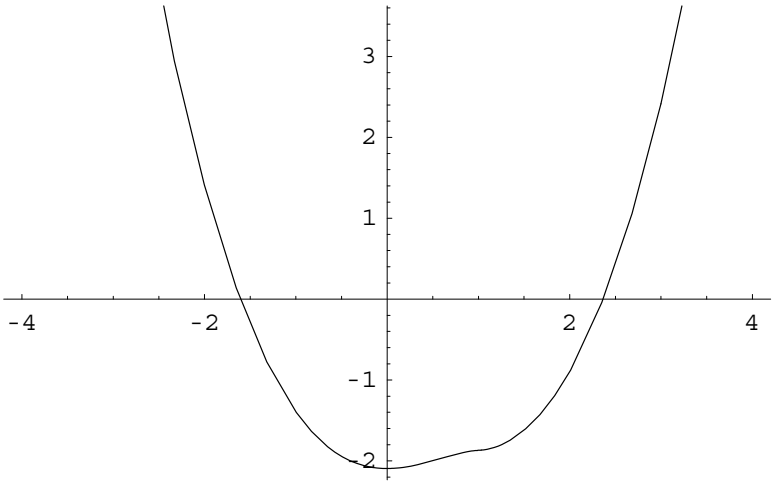
The command `Solve[eqn, var]` will attempt to find the solutions to the variable *var* in the equation *eqn*. Therefore you can use the `Solve` command to find the value of the constant `C[1]` for which `y=-2`.

```
In[14]:= const = Solve[y == -2, C[1]]
```

$$\text{Out[14]=} \quad \left\{\left\{\texttt{C[1]} \to \frac{1}{320}\ (-640 + 33\ 2^{1/3})\right\}\right\}$$

Now the specific solution is plotted by replacing `C[1]` with the constant just obtained.

In[15]:=
```
Plot[sol[[1, 1, 2]] /. C[1] → const[[1, 1, 2]], {x, -4, 4}];
```




## ■ Section 3.3 The Shape of a Graph

The following example is similar to the CAS exercise found in Section 3.3.

**Example:** Let $f(x) = \frac{a}{1+bx^2}$ with $b > 0$ and $a \neq 0$.

(a) Let $b = 1$. On a common screen, graph $f(x)$ for $a = -1,\ -0.5,\ -0.25,\ 0.25,\ 0.5$ and 1.

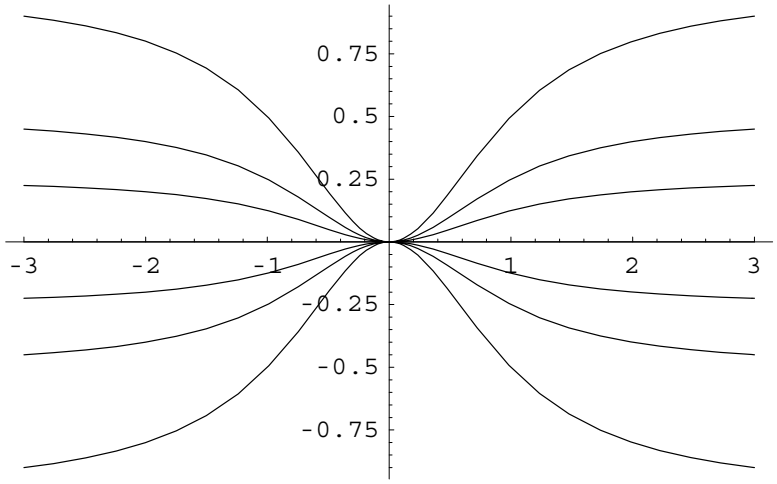
(b) Let $b = 1$. Produce a sequence of plot frames showing the graph of $y = f(x)$ for $a = -1,\ -0.9,\ -0.8,\ ...,\ 1$ and then animate the frames.

(c) Show that $f$ is decreasing on $(-\infty, 0)$ and increasing on $(0, \infty)$.

(d) Find the location of the inflection point(s) of $f$.

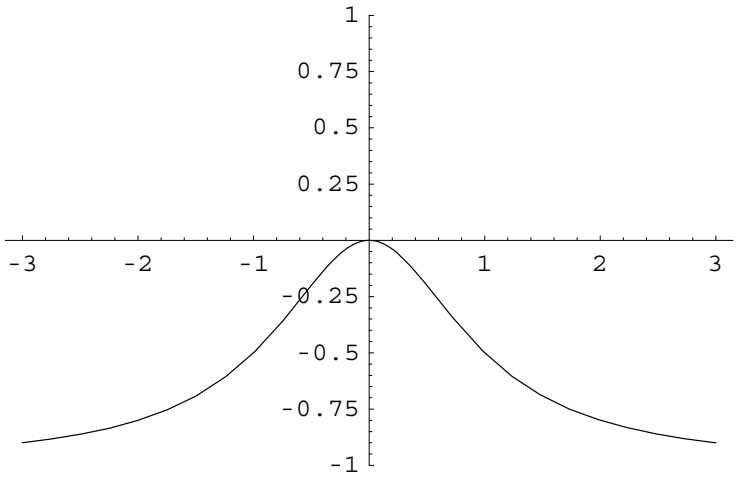
**Part (a)** The function $f$ is defined below and then the `Plot` command is used to graph the function for the specified values of $a$.

In[16]:=
```
Clear[a, b, c, f];
f[x_] = a x^2 / (1 + b x^2);
Plot[{f[x] /. {a → -1, b → 1}, f[x] /. {a → -.5, b → 1},
   f[x] /. {a → -.25, b → 1}, f[x] /. {a → 0, b → 1}, f[x] /. {a → .25, b → 1},
   f[x] /. {a → .5, b → 1}, f[x] /. {a → 1, b → 1}}, {x, -3, 3}];
```




**Part (b)** Now the frames of a movie are produced using the `Table` command (only the first frame of the movie is shown in the hard copy of this manual). After the frames have been produced, you can double-click on any frame to produce an animation. What do the graphs tell you about the number of inflections points and the concavity of the graph of $y = f(x)$?

In[19]:=
```
Table[
  Plot[f[x] /. {a → a0, b → 1}, {x, -3, 3}, PlotRange -> {-1, 1}], {a0, -1, 1, .1}];
```




**Part (c)** The first derivative is computed and simplified in the following cell. From observing the output, it is clear that $f'(x) = 0$ when $x = 0$ and since $b > 0$, the derivative is never undefined. Furthermore, if $a > 0$, then $f'(x) > 0$ when $x > 0$ and $f'(x) < 0$ when $x < 0$. Hence $f$ is increasing on $(0, \infty)$ and decreasing on $(0, \infty)$ when $a > 0$. What if $a < 0$?

```
In[20]:= Simplify[f'[x]]
```

$$\text{Out[20]=}\quad \frac{2\,a\,x}{(1+b\,x^2)^2}$$

**Part (d)** The second derivative is displayed in the following output and then the candidates for inflection points are obtained using the `Solve` command.

```
In[21]:= g[x_] = Simplify[f''[x]]
```

$$\text{Out[21]=}\quad \frac{a\,(2-6\,b\,x^2)}{(1+b\,x^2)^3}$$

```
In[22]:= Solve[g[x] == 0, x]
```

$$\text{Out[22]=}\quad \left\{\left\{x \to -\frac{1}{\sqrt{3}\ \sqrt{b}}\right\},\ \left\{x \to \frac{1}{\sqrt{3}\ \sqrt{b}}\right\}\right\}$$

To determine the inflection points and the concavity of the curve, the intervals $\left(-\infty, -\frac{1}{\sqrt{3}\ \sqrt{b}}\right)$, $\left(-\frac{1}{\sqrt{3}\ \sqrt{b}}, \frac{1}{\sqrt{3}\ \sqrt{b}}\right)$ and $\left(\frac{1}{\sqrt{3}\ \sqrt{b}}, \infty\right)$ are examined by computing the value of the second derivative at $x = \frac{-2}{\sqrt{3}\ \sqrt{b}}$, 0, and $\frac{2}{\sqrt{3}\ \sqrt{b}}$.

```
In[23]:= g[ -2 / (Sqrt[3] Sqrt[b]) ]
         g[0]
         g[ 2 / (Sqrt[3] Sqrt[b]) ]
```

$$\text{Out[23]=}\quad -\frac{162\,a}{343}$$

$$\text{Out[24]=}\quad 2\,a$$

$$\text{Out[25]=}\quad -\frac{162\,a}{343}$$

From the output, it can be seen that for a given value of $a$, the second derivative changes signs at $x = -\frac{1}{\sqrt{3}\ \sqrt{b}}$ and $x = \frac{1}{\sqrt{3}\ \sqrt{b}}$. So the points of inflection occur at $x = -\frac{1}{\sqrt{3}\ \sqrt{b}}$ and $x = \frac{1}{\sqrt{3}\ \sqrt{b}}$. If $a > 0$, where is the graph of $f$ concave up and where is it concave down? How about when $a < 0$?

## ■ Section 3.6 Linearization and Differentials

### Comparing Functions and their Linearizations

**Example:** Let $f(x) = \sqrt{x+1}$, $I = [0, 3]$ and let $a = 1$.

(a) Find the linearization $L$ of $f$ at the point $a$ and plot $f$ and $L$ on a single graph.

(b) Plot the absolute error $|f(x) - L(x)|$ over $I$ and find its maximum value.

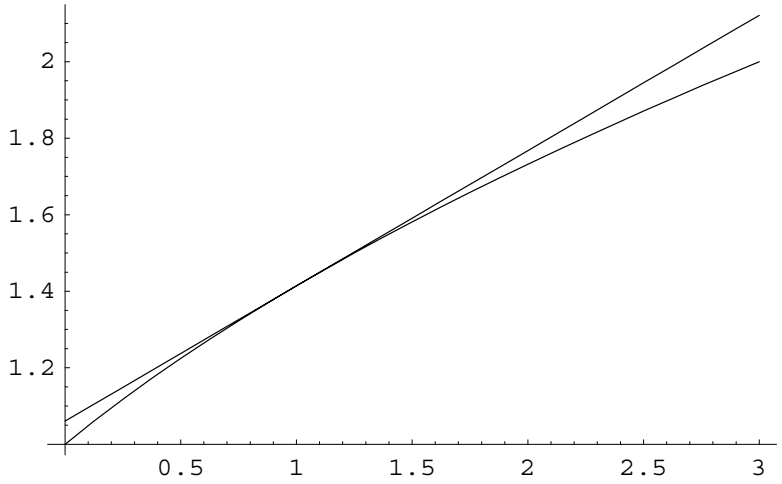
(c) For the graph in part (b), estimate as large a $\delta$>0 as possible satisfying

$$|x - a| < \delta \Rightarrow |f(x) - f(a)| < \epsilon$$

for $\epsilon = 0.01$. Then check graphically to see if your $\delta$-estimate is true.

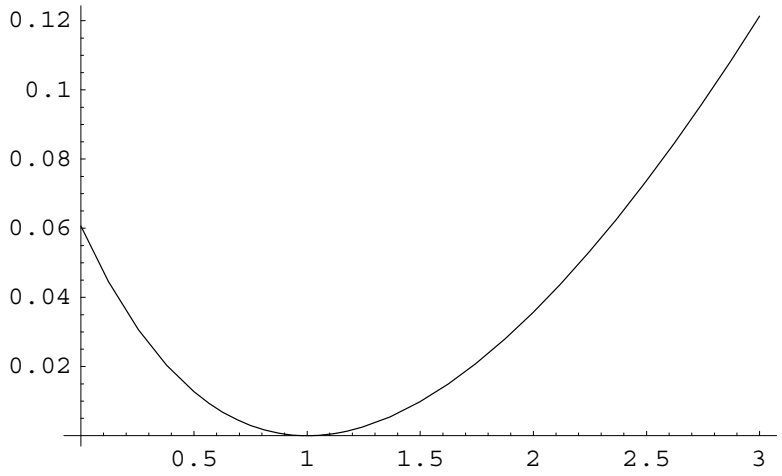
**Part (a)** The function and its corresponding linearization are given below along with a plot of their graphs.

```
In[26]:= Clear[f];
         f[x_] = √(x + 1);
         L[x_] := f'[1] (x - 1) + f[1];
         Plot[{f[x], L[x]}, {x, 0, 3}];
```



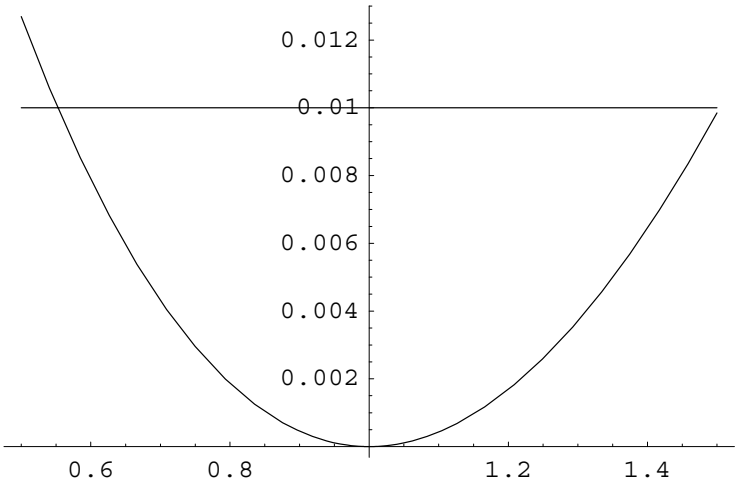

**Part (b)** The graph of $|f(x) - L(x)|$ is shown next. The *Mathematica* command `Abs[exp]` represents the absolute value of an expression *exp*.

```
In[30]:= Plot[Abs[f[x] - L[x]], {x, 0, 3}];
```



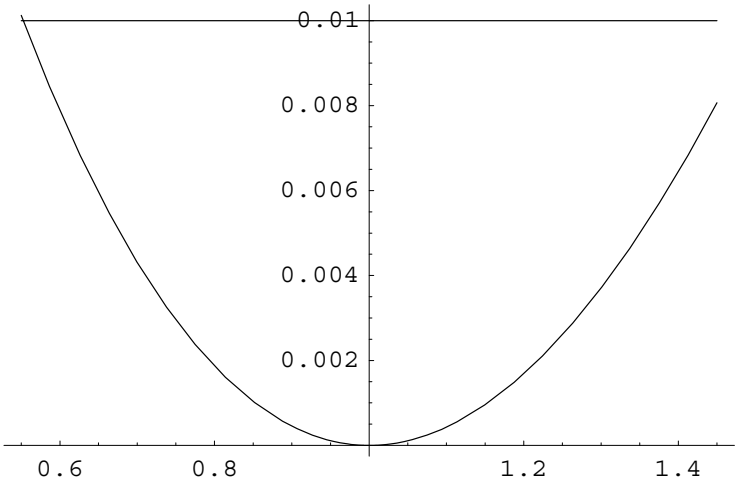

**Part (c)** From the graph given in part (b), it appears that $\delta = 0.5$ is needed in order for $|f(x) - f(a)| < 0.01$ when $|x - a| < \delta$. To see if $\delta$ is small enough, a graph of $y = |f(x) - f(a)|$ and $y = 0.01$ over the interval $|x - 1| < 0.5$ (which is equivalent to [0.5, 1.5] ) is given in the following output cell.

```
In[31]:= Plot[{Abs[f[x] - L[x]], 0.01}, {x, .5, 1.5}];
```



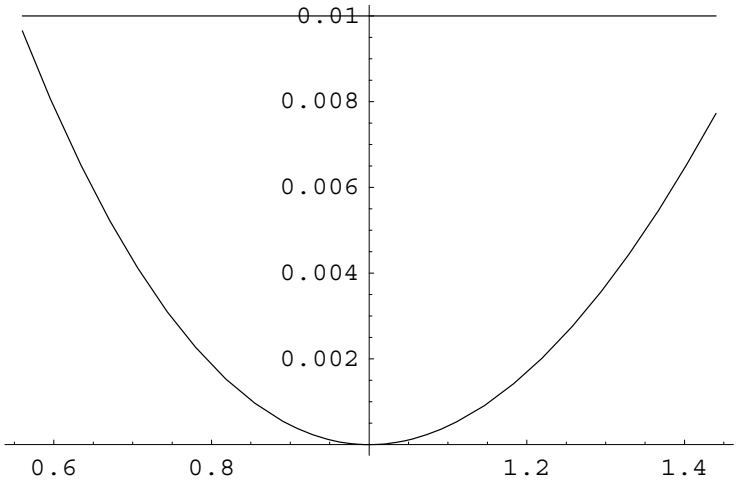

It appears that the $\delta$-estimate needs to be a little smaller. So now try $\delta = 0.45$.

```
In[32]:= Plot[{Abs[f[x] - L[x]], .01}, {x, .55, 1.45}];
```




It looks like the updated $\delta$-estimate still needs to be slightly smaller, say $\delta = .44$.

In[33]:=
```
Plot[{Abs[f[x] - L[x]], .01}, {x, .56, 1.44}];
```




So $\delta = .44$ will satisfy the following: $|x - a| < \delta \Rightarrow |f(x) - f(a)| < \epsilon$ when $\epsilon = .01$.

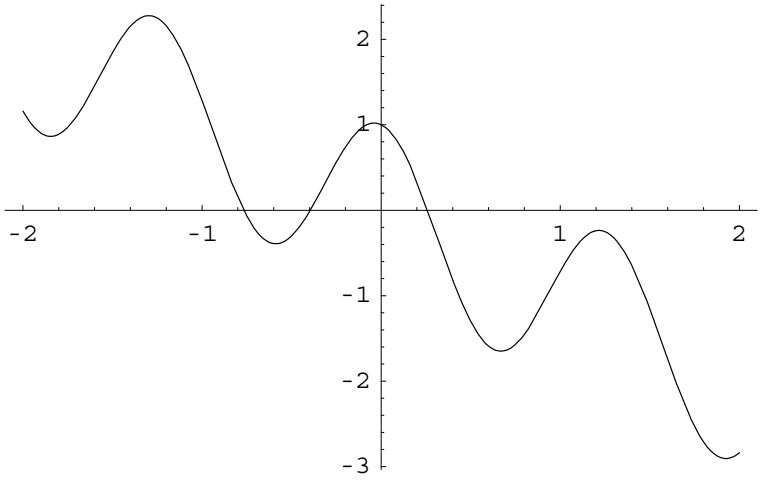
## ■ Section 3.7 Newton's Method

### Finding Roots of an $f(x)$ using Newton's Method

**Example:** Find the smallest root of $f(x) = \cos(5x) - x$ to 5 decimal places.

To find a rough approximation of the roots, the `Plot` function is used to graph $y = f(x)$ first.

In[34]:=
```
Plot[Cos[5 x] - x, {x, -2, 2}];
```




It appears that there are two negative roots - one near $-0.8$ and $-0.4$. To use Newton's method, let $g(x) = x - \frac{f(x)}{f'(x)}$.

In[35]:=
```
f[x_] = Cos[5 x] - x;
g[x_] = x - f[x] / f'[x];
```

To find the smallest root, start with $x_0 = -0.8$and then find $x_1$ using $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$which is equivalent to $x_1 = g(x_0)$.

```
In[37]:= g[-.8]
Out[37]= -0.769407
```

Therefore $x_1 = -0.769407$. Now we find $x_2$ by computing $g(x_1)$. The % symbol refers to the last output, therefore computing $g[\%]$ will yield the value of $x_2$.

```
In[38]:= g[%]
Out[38]= -0.767502
```

Now the value of $x_3$ is computed.

```
In[39]:= g[%]
Out[39]= -0.767493
```

Next, the value of $x_4$ is computed.

```
In[40]:= g[%]
Out[40]= -0.767493
```

Since the values of $x_3$ and $x_4$ are identical to 6 decimal places, the smallest root approximately equals $-0.767493$. Can you find the other two approximate roots using Newton's method?

# CAS Exercise Examples for Chapter 4: Integration

## Section 4.1 Indefinite Integrals, Differential Equations, and Modeling

### Computing Indefinite Integrals with *Mathematica*

The *Mathematica* command `Integrate[f[x],x]` will attempt to evaluate $\int f(x)\,dx$. A shortcut way of using the `Integrate` command is to open the palette entitled **BasicInput** and then click on the button containing $\int \square\, d\square$. Enter the function `f[x]`, then press the **Tab** key followed by $x$. For example, to integrate $\ln x$, the following command is executed.

```
In[1]:= ∫ Log[x] dx

Out[1]= -x + x Log[x]
```

Notice that the general form of the antiderivative is $-x + x\ln x + C$ and *Mathematica* does not include the constant $C$ in the answer above.

### Solving Initial Value Problems

**Example:** Use *Mathematica* to solve the initial value problem

$$y' = 5e^{-3x},\ \ y(0) = -10.$$

Begin by letting `y[x]` equal the antiderivative and add a `c` to the answer.

```
In[2]:= y[x_] = ∫ 5 e^(-3 x) dx + c

Out[2]= c - 5 E^(-3 x)/3
```

The `Solve` command is now used to find the value of $c$.

```
In[3]:= cval = Solve[y[0] == -10, c]

Out[3]= {{c → -25/3}}
```

Now the command `cval[[1,1,2]]` is used to extract the value of $c$ in the following assignment statement.

```
In[4]:= c = cval[[1, 1, 2]]

Out[4]= -25/3
```

The `Factor` command can be used to factor the value of $y(x)$.

```
In[5]:= Factor[y[x]]
```

$$\text{Out[5]}= -\frac{5}{3}\,E^{-3x}\,(1+5\,E^{3x})$$

Here is a slightly more challenging example.

**Example:** Solve the initial value problem

$$y'' = e^{2x} + \tfrac{1}{x^2},\ \ y(1) = 2,\ \ y'(1) = 4$$

Study the following steps used to find the solution.

```
In[6]:= yprime[x_] = ∫ (e^-2x + 1/x^2) dx + c1
```

$$\text{Out[6]}= c1 - \frac{E^{-2x}}{2} - \frac{1}{x}$$

```
In[7]:= sol = Solve[yprime[1] == 4, c1]
```

$$\text{Out[7]}= \left\{\left\{c1 \to 5 + \frac{1}{2\,E^2}\right\}\right\}$$

```
In[8]:= c1 = sol[[1, 1, 2]]
```

$$\text{Out[8]}= 5 + \frac{1}{2\,E^2}$$

```
In[9]:= Clear[y];
        y[x_] = ∫ yprime[x] dx + c2
```

$$\text{Out[9]}= c2 + \frac{E^{-2x}}{4} + \frac{(1+10\,E^2)\,x}{2\,E^2} - \text{Log}[x]$$

```
In[10]:= sol = Solve[y[1] == 2, c2]
```

$$\text{Out[10]}= \left\{\left\{c2 \to -\frac{3\,(1+4\,E^2)}{4\,E^2}\right\}\right\}$$

```
In[11]:= c2 = sol[[1, 1, 2]]
```

$$\text{Out[11]}= -\frac{3\,(1+4\,E^2)}{4\,E^2}$$

Here is the solution:

```
In[12]:= y[x]
```

$$\text{Out[12]}= \frac{E^{-2x}}{4} - \frac{3\,(1+4\,E^2)}{4\,E^2} + \frac{(1+10\,E^2)\,x}{2\,E^2} - \text{Log}[x]$$

## ■ Section 4.3 Estimating with Finite Sums

The following example is similar to Exercises 25 - 28 in your text.

**Example:** Let $f(x) = x^3 + \sin x$ on $[1, \pi]$ and use *Mathematica* to perform the following steps.

(a) Plot the function over the given interval.

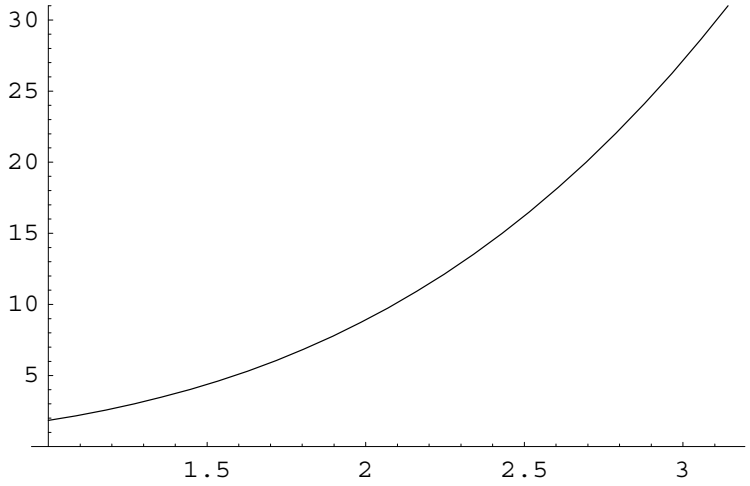
(b) Partition the given interval into 100 subintervals of equal length and evaluate the function at the midpoint of each subinterval.

(c) Compute the average value of the function values generated in part (b).

(d) Solve the equation $f(x)$ = (average value) for $x$ using the average value calculation in part (c).

**Part (a)** First, the function is defined and graphed over the given interval.

```
In[13]:= f[x_] = x^3 + Sin[x];
         Plot[f[x], {x, 1, π}, PlotRange -> {0, π^3 + Sin[π]}];
```




**Part (b)** The value of Δx is defined below where the greek letter Δ was entered from the **BasicInput** palette. Then the `Table` command is used to partition the interval and to evaluate the function at the midpoint of each subinterval.

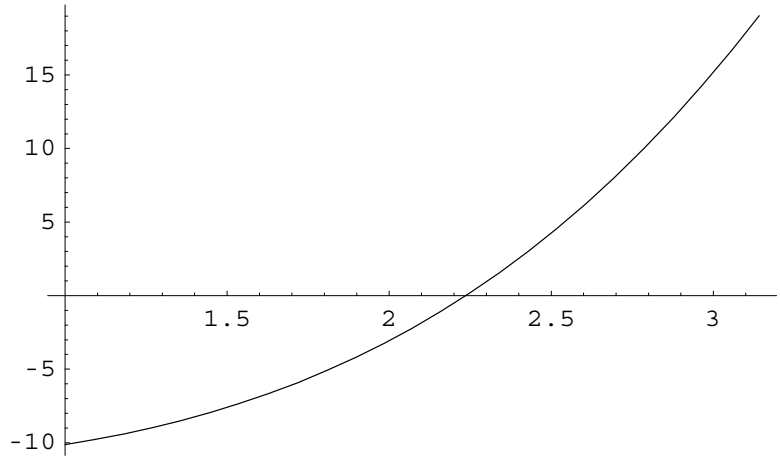
```
In[14]:= Δx = (π - 1)/100;
         fvals = Table[f[1 + (i - .5) Δx], {i, 1, 100}]
Out[14]= {1.87968, 1.95789, 2.03855, 2.12171, 2.20743, 2.29575, 2.38674, 2.48045, 2.57693,
          2.67624, 2.77843, 2.88356, 2.99169, 3.10286, 3.21715, 3.33459, 3.45526,
          3.5792, 3.70648, 3.83715, 3.97126, 4.10888, 4.25007, 4.39487, 4.54335,
          4.69558, 4.85159, 5.01147, 5.17525, 5.34301, 5.5148, 5.69069, 5.87072,
          6.05497, 6.24349, 6.43635, 6.63359, 6.8353, 7.04151, 7.25231, 7.46774,
          7.68787, 7.91277, 8.14249, 8.37709, 8.61665, 8.86121, 9.11085, 9.36563,
          9.62561, 9.89085, 10.1614, 10.4374, 10.7188, 11.0057, 11.2983, 11.5964,
          11.9003, 12.21, 12.5255, 12.8469, 13.1743, 13.5077, 13.8472, 14.193, 14.5449,
          14.9031, 15.2678, 15.6388, 16.0164, 16.4005, 16.7912, 17.1887, 17.5929,
          18.004, 18.422, 18.8469, 19.2789, 19.718, 20.1642, 20.6177, 21.0786, 21.5467,
          22.0224, 22.5055, 22.9963, 23.4947, 24.0008, 24.5147, 25.0364, 25.5661,
          26.1037, 26.6495, 27.2033, 27.7654, 28.3357, 28.9143, 29.5014, 30.0969, 30.701}
```

**Part (c)** For a given list *lis*, the *Mathematica* command `Plus@@lis` will compute the sum of all the numbers in the *lis*.

```
In[15]:= averageval = (Plus @@ fvals) / 100
Out[15]= 11.9734
```

**Part (d):** Solving $f(x)$ = (average value) for $x$ is equivalent to solving $f(x)$ − average value = 0 for $x$. You can plot the $f(x)$ − (average value) first to obtain the approximate location of the root and then use the `FindRoot` command to obtain the root.

```
In[16]:= Plot[f[x] - averageval, {x, 1, π}];
```




```
In[17]:= r = FindRoot[f[x] - averageval, {x, 2.2}]
Out[17]= {x → 2.23651}
```

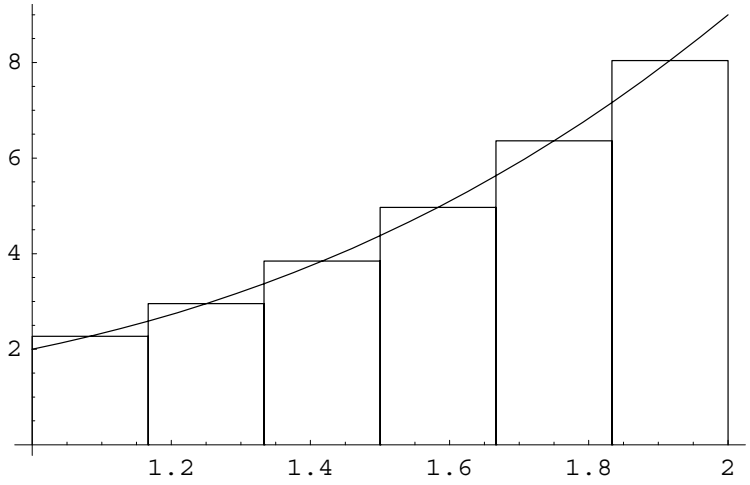
## ■ Section 4.4 Riemann Sums and Definite Integrals

In Exercises 41 - 46, you are asked to draw rectangles associated with Riemann sums. There is no built-in *Mathematica* function which will do this, but by executing the following command, you will create a function `RiemannSum[a, b, n, m]` which will illustrate and compute the sum $\sum_{k=1}^{n} f(c_k)\,\Delta x$ where $c_k$ is the midpoint of the corresponding subinterval if $m = 0.5$.

```
In[18]:= RiemannSum[a_, b_, n_, m_] := Module[{deltax = (b - a)/n, recs, fungraph, intest},
            intest = deltax Sum[f[a + (i - 1 + m) deltax], {i, 1, n}];
            recs = ListPlot[Flatten[
          Table[{{a + (i - 1) deltax, 0}, {a + (i - 1) deltax, f[a + (i - 1 + m) deltax]},
            {a + i deltax, f[a + (i - 1 + m) deltax]}, {a + i deltax, 0}}, {i, 1, n}], 1],
        PlotJoined -> True, DisplayFunction -> Identity, PlotRange -> All];
            fungraph = Plot[f[x], {x, a, b}, DisplayFunction -> Identity,
        PlotRange -> All];
            Show[{recs, fungraph}, DisplayFunction -> $DisplayFunction];
            Return[intest]]
```

This newly created command is now used in the following example.

```
In[19]:= f[x_] := x^3 + 1;
         RiemannSum[1, 2, 6, .5]
```




```
Out[19]= 4.73958
```

The value of $m$ in `RiemannSum[a, b, n, m]` can be any value between 0 and 1 inclusive. Repeat the command above, but try different values of $m$ and describe what is happening. For example, let $m = 1$ and observe how the rectangles are formed. What happens if you use $m = 0$?

## ■ Section 4.5 The Mean Value and Fundamental Theorems

### Computing Definite Integrals with *Mathematica*

The *Mathematica* command `Integrate[f[x],{x,a,b}]` will attempt to evaluate $\int_a^b f(x)\,dx$. A shortcut way of using the `Integrate` command is to open the palette entitled **BasicInput** and then click on the button containing $\int_{\square}^{\square} \square\, d\square$.

```
In[20]:= Integrate[Log[x], {x, 1, 2}]
Out[20]= -1 + Log[4]
```

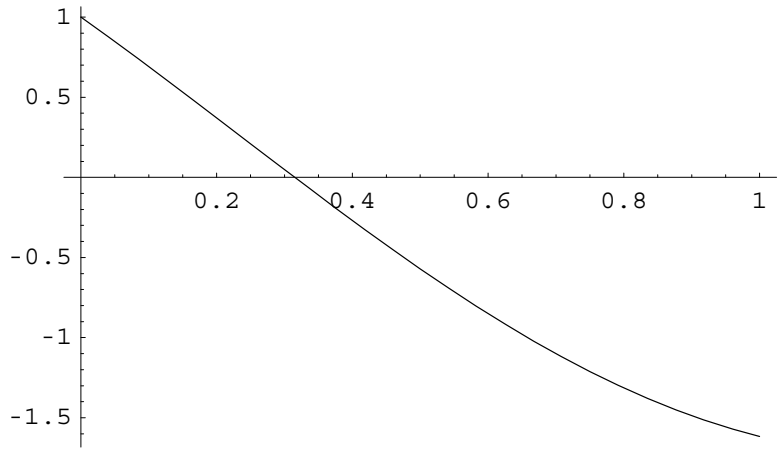
**Example:** Let $f(x) = -x - \sin(2x) + \cos^2 x$ on $[a, b] = [0, 1]$ and use *Mathematica* to perform the following steps. Let $F(x) = \int_a^x f(x)\,dx$ and solve $F'(x) = 0$.

The solution to this example is straight forward. First the function $F$ is defined.

```
In[21]:= Clear[f];
         f[x_] = -x - Sin[2 x] + Cos[x]^2;
         F[x_] = Integrate[f[t], {t, 0, x}]
Out[21]= -1/2 + 1/4 (2 x - 2 x^2 + 2 Cos[2 x] + Sin[2 x])
```

Now the approximate location of the root $F'(x)$ is found after first graphing the function to determine the approximate location of the root.

```
In[22]:= Plot[F'[x], {x, 0, 1}];
```




```
In[23]:= FindRoot[F'[x], {x, .3}]
Out[23]= {x → 0.314959}
```

## ■ Section 4.6: Substitution in Definite Integrals

**Example:** Let $f(x) = \sin^2 x$ and $g(x) = x(x-1)(x-3)(x-4)$. Use *Mathematica* to complete the following steps.

(a) Plot the curves together to determine the number of points of intersection.

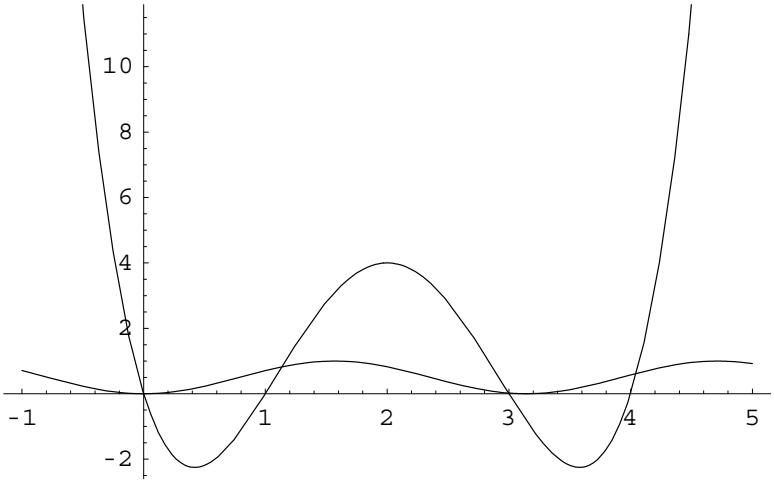
(b) Determine where the curves intersect.

(c) Integrate $|f(x) - g(x)|$ over consecutive pairs of intersection values.

(d) Sum together the integrals found in part (c).

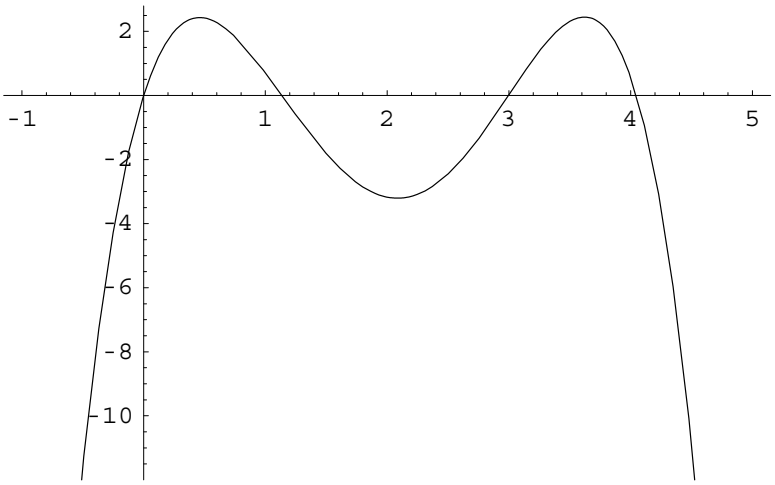
**Part (a)** Study the following input and output. How many points of intersection do you see?

```
In[24]:= Clear[f, g];
         f[x_] = Sin[x]^2;
         g[x_] = x (x - 1) (x - 3) (x - 4);
         Plot[{f[x], g[x]}, {x, -1, 5}];
```




**Part (b)** Solving $f(x) = g(x)$ is equivalent to solving $f(x) - g(x) = 0$. By plotting $f(x) - g(x)$, you can see the approximate locations of the roots. One of the roots is 0 (why?) and the remaining roots can be found using the `FindRoot` command.

In[25]:=
```
Plot[f[x] - g[x], {x, -1, 5}];
```




In[26]:=
```
int2 = FindRoot[f[x] - g[x], {x, 1.1}]
int3 = FindRoot[f[x] - g[x], {x, 3}]
int4 = FindRoot[f[x] - g[x], {x, 4}]
```

Out[26]= {x → 1.13558}

Out[27]= {x → 2.99652}

Out[28]= {x → 4.04794}

**Parts (c) and (d)** Now you can integrate $|f(x) - g(x)|$ over consecutive pairs of intersection values and sum up the values obtained.

In[29]:=
```
area1 = ∫_0^int2[[1,2]] (f[x] - g[x]) dx
```

Out[29]= 1.78766

In[30]:=
```
area2 = ∫_int2[[1,2]]^int3[[1,2]] (g[x] - f[x]) dx
```

Out[30]= 3.81784

In[31]:=
```
area3 = ∫_int3[[1,2]]^int4[[1,2]] (f[x] - g[x]) dx
```

Out[31]= 1.6636

In[32]:=
```
area1 + area2 + area3
```

Out[32]= 7.26909

## ■ Section 4.7 Numerical Integration

### Numerical Integration

The *Mathematica* command `NIntegrate[f[x],{x,a,b}]` will numerically integrate $\int_a^b f(x)\,dx$.

```
In[33]:= NIntegrate[e^-x^2, {x, -1, 1}]
Out[33]= 1.49365
```

If you want a command that will estimate $\int_a^b f(x)\,dx$ with the Trapezoidal rule for a given value of $n$, then execute the following command to create the function. A picture is given illustrating the trapezoids formed by the Trapezoidal rule. The command `TrapPic[a, b, n]` will use $n$ subintervals to estimate $\int_a^b f(x)\,dx$

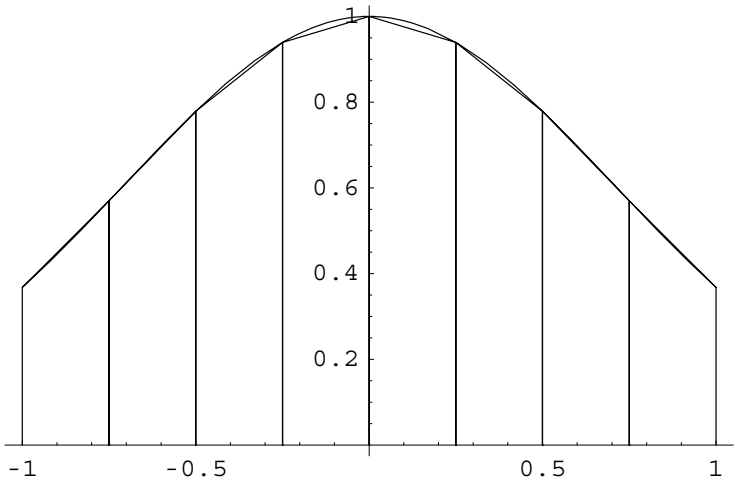
```
In[34]:= TrapPic[a_, b_, n_] := Module[{h = (b - a)/n, traps, fungraph, intest},
           intest = (h/2) (f[a] + 2 Sum[f[a + i h], {i, 1, n - 1}] + f[b]);
           traps = ListPlot[Flatten[Table[{{a + i h, 0}, {a + i h, f[a + i h]},
             {a + (i + 1) h, f[a + (i + 1) h]}, {a + (i + 1) h, 0}}, {i, 0, n - 1}], 1],
         PlotJoined -> True, DisplayFunction -> Identity, PlotRange -> All];
           fungraph = Plot[f[x], {x, a, b}, DisplayFunction -> Identity,
         PlotRange -> All];
           Show[{traps, fungraph}, DisplayFunction -> $DisplayFunction];
           Return[intest]]
```

If you don't care to see a picture, use the following command instead.

```
In[35]:= TrapRule[a_, b_, n_] := Module[{h = (b - a)/n, intest},
           intest = (h/2) (f[a] + 2 Sum[f[a + i h], {i, 1, n - 1}] + f[b]);
           Return[intest]]
```

Here is an example.

In[36]:=
```
Clear[f];
f[x_] = e^-x^2;
N[TrapPic[-1, 1, 8]]
```




Out[36]= 1.48597

Here is the same thing without a displayed picture

In[37]:=
```
N[TrapRule[-1, 1, 8]]
```

Out[37]= 1.48597

# CAS Exercise Examples for Chapter 5: Applications of Integration

---

## ■ Section 5.3 Lengths of Plane Curves

### Determining the Arc Length of a Smooth Curve

To determine the length of a curve, you can first try to integrate using $\int_a^b \sqrt{1 + f'[x]^2}\, dx$ and if this fails to produce an answer, then use the numerical integration command `NIntegrate[`$\sqrt{1 + f'[x]^2}$`,{x,a,b}]`. In the following example, both commands produce the desired result.

In[1]:=
$$f[x\_] = \frac{4\sqrt{2}}{3} x^{\frac{3}{2}} - 1;$$

$$\int_0^1 \sqrt{1 + f'[x]^2}\, dx$$

Out[1]=
$$\frac{13}{6}$$

In[2]:=
$$\text{NIntegrate}\left[\sqrt{1 + f'[x]^2}, \{x, 0, 1\}\right]$$

Out[2]= `2.16667`

### Comparing the Length of the Polygonal Path with the Length of a Curve

The following example is similar to Exercises 31 - 36.

**Example:** Let $f(x) = x^3$, $0 \le x \le 2$.

(a) Plot the curve together with the polygonal path approximation for $n = 3$.

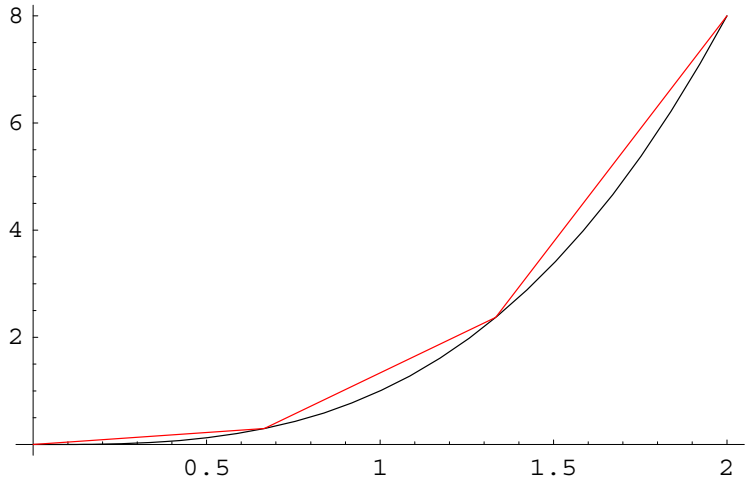
(b) Find the corresponding approximate length of the curve by summing the lengths of the line segments and then evaluate the actual length using an integral.

**Part (a)** Given that $\Delta x$ is defined as $\frac{b-a}{n}$ where $a = 0$, $b = 2$ and $n = 3$, the command

```
Table[{a+i △x,f[a+i △x]},{i,0,n}]
```

will form a table of ordered pairs representing the endpoints of each straight line in the polygonal path. You can then use the `ListPlot` command to plot the polygonal path. Study the following input and output.

```
In[3]:= f[x_] = x^3;
        a = 0; b = 2; n = 3;
        Δx = (b - a)/n;
        pts = Table[{a + i Δx, f[a + i Δx]}, {i, 0, n}];
        polypath = ListPlot[pts, PlotJoined -> True, PlotRange -> All,
           DisplayFunction -> Identity, PlotStyle -> RGBColor[1, 0, 0]];
        fungraph = Plot[f[x], {x, a, b}, DisplayFunction -> Identity, PlotRange -> All];
        Show[{fungraph, polypath}, DisplayFunction -> $DisplayFunction];
```




Part (b) The command `Table[Sqrt[Δx^2 + (f[a + i Δx] - f[a + (i - 1) Δx])^2], {i, 1, n}]` will form a table whose entries are the lengths of each line segment making up the polygonal path. Then we can add these lengths together using `Plus@@`.

```
In[4]:= Plus @@ Table[Sqrt[Δx^2 + (f[a + i Δx] - f[a + (i - 1) Δx])^2], {i, 1, n}] // N
Out[4]= 8.57709
```

Compare this value to the actual length of the curve.

```
In[5]:= NIntegrate[Sqrt[1 + f'[x]^2], {x, 0, 2}]
Out[5]= 8.63033
```

# CAS Exercise Examples for Chapter 6: Transcendental Functions and Differential Equations

---

## Section 6.2 Exponential Functions (Section 2.8 Derivatives of Inverse Trigonometric Functions in Early Transcendentals Version)

### Plotting Functions and their Inverses on One Graph

The purpose of the following example is it explore a function and its inverse

together with their derivatives.

**Example:** Let $f(x) = \cos x$, $0 \le x \le \frac{\pi}{2}$ and $x_0 = \frac{\pi}{6}$. Complete each of the following with *Mathematica*.

a) Plot $y = f(x)$ and $y = f\,'(x)$ together on one graph.

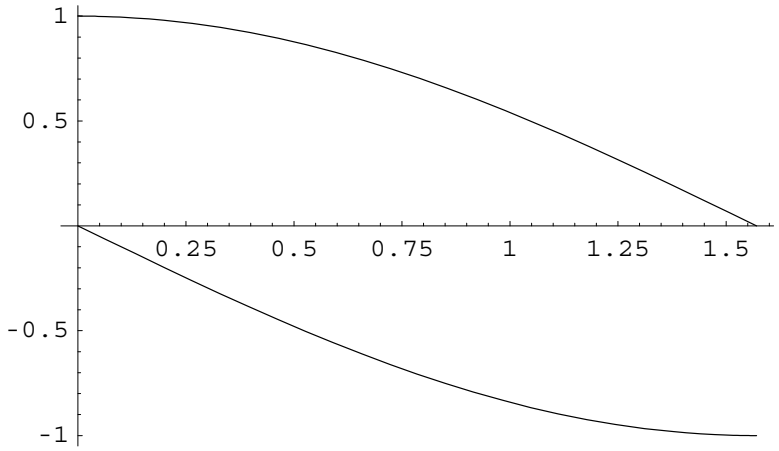
b) Find the equation of the line tangent to $f$ at $x = x_0$. Plot the tangent line together with $f$.

c) Find the inverse (call it $g$) of $f$ and then find the line tangent to $g$ at $(f(x_0), x_0)$. (Use Theorem 1.)

d) Plot $f$, $g$, the two tangent lines, the identity and the line segment joining the points $(x_0, f(x_0))$ and $(f(x_0), x_0)$.

**Part (a)** Study the following input and output.

```
In[1]:= f[x_] = Cos[x];
        Plot[{f[x], f'[x]}, {x, 0, π / 2}];
```

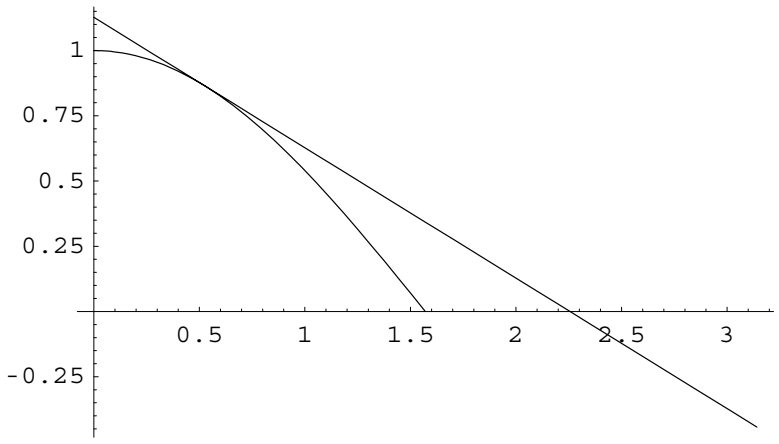



**Part (b)** The following two input cells show how to create a tangent line along with the graph of the tangent line and the function.

```
In[2]:= L[x_] = f[π / 6] + f'[π / 6] (x - π / 6)
```

$$\text{Out[2]=}\quad \frac{\sqrt{3}}{2} + \frac{1}{2}\left(\frac{\pi}{6} - x\right)$$

```
In[3]:= fplot = Plot[f[x], {x, 0, π / 2}, DisplayFunction -> Identity];
        ftanplot = Plot[L[x], {x, 0, π}, DisplayFunction -> Identity];
        Show[{fplot, ftanplot}, DisplayFunction -> $DisplayFunction];
```




**Part (c)** The `Solve` command is now used to find the inverse of $f$ (ignore the warning message in the following output).

```
In[4]:= invsol = Solve[y == f[x], x]
        Solve::ifun : Inverse functions are
           being used by Solve, so some solutions may  not be found.
Out[4]= {{x → -ArcCos[y]}, {x → ArcCos[y]}}
```

A function $g$ is now created for the inverse.

```
In[5]:= g[y_] = invsol[[2, 1, 2]];
        g[x]
Out[5]= ArcCos[x]
```

The tangent line to the inverse is now created.

```
In[6]:= m = 1 / f'[π / 6];
        LInv[x_] = π / 6 + m (x - f[π / 6]);
```

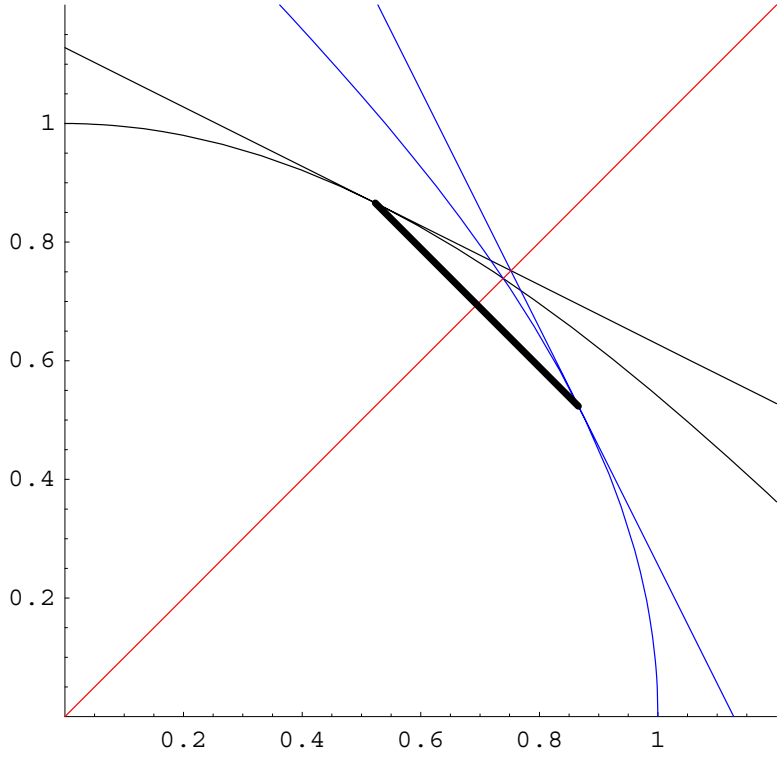
The inverse, its tangent line and the identity $y = x$ are all plotted for later use. Ignore the warning messages.

```
In[7]:= gplot = Plot[g[x], {x, 0, 1},
           PlotStyle -> RGBColor[0, 0, 1], DisplayFunction -> Identity];
        gtanplot = Plot[LInv[x], {x, 0, 2}, PlotStyle -> RGBColor[0, 0, 1],
           DisplayFunction -> Identity];
        identityplot = Plot[x, {x, 0, 2}, PlotStyle -> RGBColor[1, 0, 0],
           DisplayFunction -> Identity];
        General::spell1 : Possible spelling error: new
           symbol name "gplot" is similar to  existing symbol "fplot".
        General::spell1 : Possible spelling error: new symbol
           name "gtanplot" is similar  to existing symbol "ftanplot".
```

The line segment joining the points $(x_0,\ f(x_0))$ and $(f(x_0),\ x_0)$ is now created and then the required graph is displayed.

```
In[8]:= segment = ListPlot[{{π / 6, f[π / 6]}, {f[π / 6], π / 6}}, PlotJoined -> True,
           PlotStyle -> Thickness[.01], DisplayFunction -> Identity];
```

```
In[9]:= Show[{fplot, ftanplot, gplot, gtanplot, identityplot, segment},
          PlotRange -> {{0, 1.2}, {0, 1.2}}, AspectRatio -> 1,
          DisplayFunction -> $DisplayFunction];
```




## Section 6.4 First-Order Separable Differential Equations (Section 5.4 in Early Transcendentals Version)

### Slope Fields

A package called **PlotField**, contained in a file called **Graphics,** contains a command called `PlotVectorField` which can be used to plot the slope field of a differential equation.

```
In[10]:= << Graphics`PlotField`
```

**Example:** Obtain a slope field for the differential equation $y' = \sqrt{xy}$ and a graph of the solution curve passing through (0, 1).

First, the solution is found by first noting that the differential equation is equivalent to $\frac{1}{\sqrt{y}}\,dy = \sqrt{x}\,dx$. So you can begin by assigning an arbitrary name (such as `eq`) to the equation after the antiderivative of each side has been computed.

```
In[11]:= eq = ∫ 1 / √y dy == ∫ √x dx + c
```

$$\text{Out[11]=}\quad 2\sqrt{y} == c + \frac{2\,x^{3/2}}{3}$$

Now the value of $c$ is obtained using the `Solve` and `/.` commands.

```
In[12]:= cval = Solve[eq /. {x → 0, y → 1}, c]
Out[12]= {{c → 2}}

In[13]:= cval[[1, 1, 2]]
Out[13]= 2
```

The Solve command is used again to find the solution for $y$ in terms of $x$.

```
In[14]:= sol = Solve[eq /. {c → cval[[1, 1, 2]]}, y]
Out[14]= {{y → 1/9 (9 + 6 x^(3/2) + x^3)}}
```

Now you can create a function $y = f(x)$ representing the solution just found.

```
In[15]:= Clear[f];
         f[x_] = sol[[1, 1, 2]]
Out[15]= 1/9 (9 + 6 x^(3/2) + x^3)
```

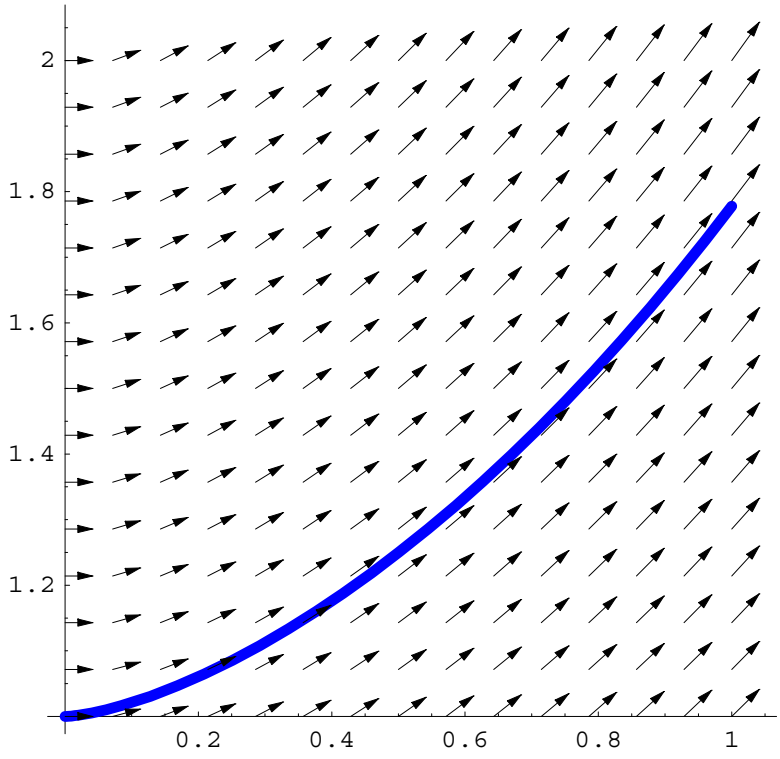
The *Mathematica* command `PlotVectorField[{1,f[x,y]},{x,xmin,xmax},{ymin,ymax}]` will plot the slope field of the differentiable equation $y' = f(x, y)$. The option `AspectRatio->1` makes the graph height to width ratio equal to one. The option

`PlotStyle -> {Thickness[.015], RGBColor[0, 0, 1]}`

is used to increase the thickness of the solution curve and to display the curve in color blue.

```
In[16]:= dirfield = PlotVectorField[{1, Sqrt[x y]}, {x, 0, 1},
           {y, 1, 2}, AspectRatio -> 1, DisplayFunction -> Identity];

In[17]:= solgraph = Plot[f[x], {x, 0, 1}, AspectRatio -> 1, PlotStyle ->
            {Thickness[.015], RGBColor[0, 0, 1]}, DisplayFunction -> Identity];
```

In[18]:= `Show[{solgraph, dirfield}, DisplayFunction -> $DisplayFunction];`




## Section 6.6 Euler's Method; Population Models (Section 6.4 in Early Transcendentals Version)

### Using *Mathematica* to Perform Euler's Method

Using *Mathematica*, you can define the functions `x[i]` to represent $x_i$, `y[i]` to represent $y_i$ and `f[x,y]` to equal $f(x, y)$. Then, after defining x[0] and y[0] to equal the given values of $x_0$ and $y_0$ and setting dx to equal the step size, let

`x[i_]:=x[i-1] + dx` and `y[i_]:=y[i-1] + f[x[i-1],y[i-1]] dx`

Notice also that delayed equals (:=) are used for these two assignment statements since we do not want to evaluate the right hand side of the equations until a specific value of *i* is used.

When *Mathematica* is then asked to compute `y[3]` for example, it will first compute `y[0]`, `y[1]` and `y[2]` using the assignment statements above. If you then have *Mathematica* compute `y[4]`, it will first recompute all the previous values `y[0]` through `y[3]` *again*. To let *Mathematica* remember the previous values of x and y, use the following assignment statements instead:

`x[i_]:=x[i]=x[i-1]+dx` and

`y[i_]:=y[i]=y[i-1]+f[x[i-1],y[i-1]] dx`

Consider the following example.

**Example:** Use Euler's method to estimate the value of the solution to the initial value problem $y' = 1 - x\,e^{y}$, $y(0) = 1$ at the point $x^{*} = 2$ using the step size dx = 0.1. Compare the accuracy of your solution with the exact solution.

Here are the necessary assignment statements for Euler's method.

```
In[19]:= Clear[f, x, y];
         f[x_, y_] = 1 + x^2 y;
         x[0] = 1.; y[0] = 2.; dx = 0.1;
         x[i_] := x[i] = x[i - 1] + dx
         y[i_] := y[i] = y[i - 1] + f[x[i - 1], y[i - 1]] dx
```

The command `Table[{x[i],y[i]},{i,0,n}]` will form $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$.

```
In[21]:= eulervals = Table[{x[i], y[i]}, {i, 0, 10}]

Out[21]= {{1., 2.}, {1.1, 2.3}, {1.2, 2.6783}, {1.3, 3.16398},
          {1.4, 3.79869}, {1.5, 4.64323}, {1.6, 5.78796},
          {1.7, 7.36967}, {1.8, 9.59951}, {1.9, 12.8097}, {2., 17.5341}}
```

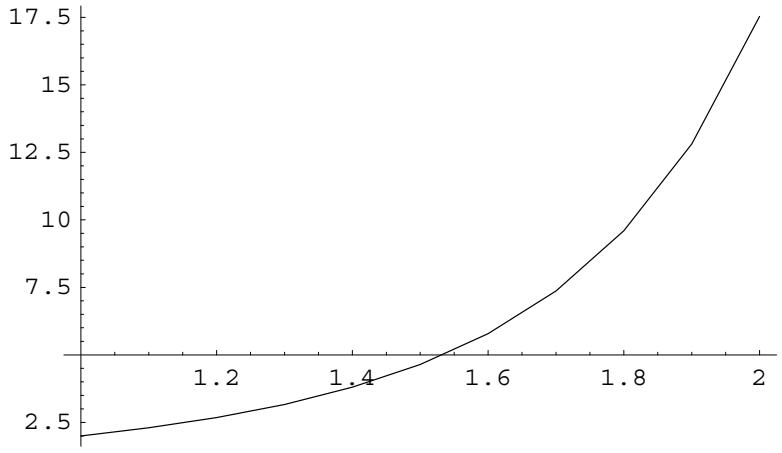
A nice command for better visualization of the list of order pairs is the `TableForm` command and the additional option `TableHeadings`, appearing in the following input cell, will place appropriate headings on each column.

```
In[22]:= TableForm[eulervals, TableHeadings -> {None, {"x_i", "y_i"}}]

Out[22]//TableForm=
x_i     y_i
1.      2.
1.1     2.3
1.2     2.6783
1.3     3.16398
1.4     3.79869
1.5     4.64323
1.6     5.78796
1.7     7.36967
1.8     9.59951
1.9     12.8097
2.      17.5341
```

From the table, it appears that $y \approx 17.5341$ when $x = 2$.

If you want, you can now take the ordered pairs in the table and plot them to get an approximation of the solution curve.

In[23]:=
```
eulerplot = ListPlot[eulervals, PlotJoined -> True];
```




The exact solution to $y' = f(x, y)$ where $y(x_0) = y_0$ can be found using the command

`DSolve[{y'[x] == f[x, y], y[x0] == y0}, y[x], x].`

In[24]:=
```
Clear[x, y];
sol = DSolve[{y'[x] == 1 + x^2 y[x], y[1] == 2}, y[x], x]
```

Out[24]=
$$\left\{\left\{y[x] \to \frac{E^{\frac{x^3}{3}}\left(3\,(x^3)^{1/3}\left(\frac{2}{E^{1/3}} + \frac{\text{Gamma}\left[\frac{1}{3},\frac{1}{3}\right]}{3^{2/3}}\right) - 3^{1/3}\,x\,\text{Gamma}\left[\frac{1}{3},\frac{x^3}{3}\right]\right)}{3\,(x^3)^{1/3}}\right\}\right\}$$

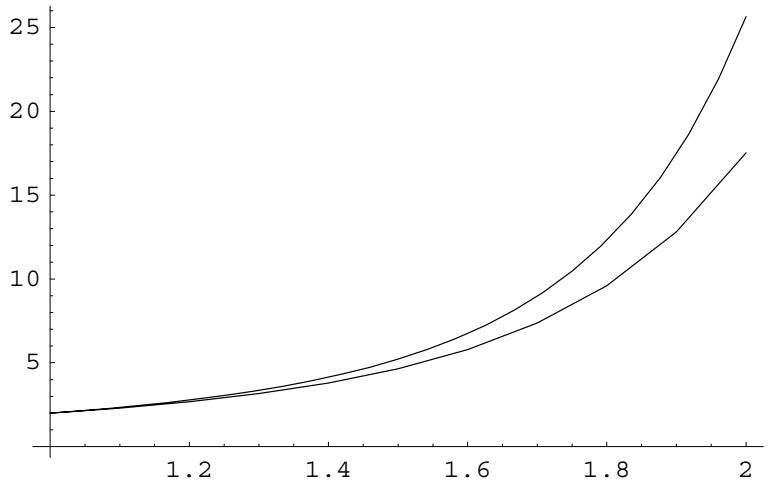
Now the exact value of $y$ when $x = 2$ is found.

In[25]:=
```
exactyval = sol[[1, 1, 2]] /. x -> 2.
```

Out[25]=
```
25.6527
```

To see why our approximation is so far off, a plot of the approximate solution and the exact solution are displayed in the following output cell. To obtain a better approximation, a smaller value of dx should be used.

```
In[26]:= exactplot = Plot[sol[[1, 1, 2]], {x, 1, 2}, DisplayFunction -> Identity];
         Show[eulerplot, exactplot];
```




### Improved Euler's Method

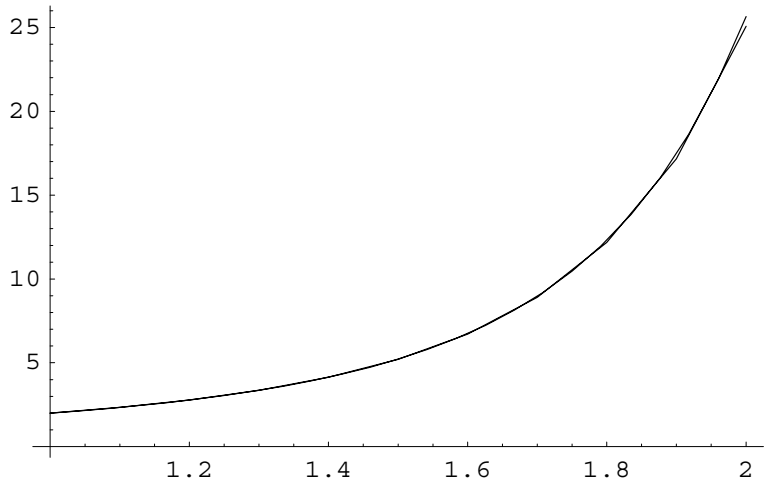
If you replace the previous code for Euler's method with the following improved Euler's method, much better results will usually be obtained. Study the following input cell.

```
In[27]:= Clear[f, x, y];
         f[x_, y_] = 1 + x^2 y;
         x[0] = 1.; y[0] = 2.; dx = 0.1;
         x[i_] := x[i] = x[i - 1] + dx;
         z[i_] := z[i] = y[i - 1] + f[x[i - 1], y[i - 1]] dx;
         y[i_] := y[i] = y[i - 1] + (f[x[i - 1], y[i - 1]] + f[x[i], z[i]])/2 dx;

In[28]:= improvedEulervals = Table[{x[i], y[i]}, {i, 0, 10}]

Out[28]= {{1., 2.}, {1.1, 2.33915}, {1.2, 2.77667},
          {1.3, 3.35345}, {1.4, 4.1308}, {1.5, 5.20266}, {1.6, 6.71654},
          {1.7, 8.9097}, {1.8, 12.1739}, {1.9, 17.1734}, {2., 25.0678}}
```

From the table, it appears that $y \approx 25.0678$ when $x=2$. Much better! The plot of the graphs show how good the approximate solution is compared to the exact solution.

```
In[29]:= improvedeplot =
           ListPlot[improvedEulervals, PlotJoined -> True, DisplayFunction -> Identity];
```

In[30]:=
```
Show[{improvedeplot, exactplot}, DisplayFunction -> $DisplayFunction];
```



# CAS Exercise Examples for Chapter 7: Integration Techniques, L'Hôpital's Rule and Improper Integrals

## ■ Section 7.5 Integral Tables, Computer Algebra Systems, and Monte Carlo Integration

### Integration with a Computer Algebra System

The command `Integrate[f[x],x]` or $\int f[x]\,dx$ can be used to find indefinite integrals. See the following examples.

*In[1]:=*
$$\int x\,e^x\,dx$$
$$\int x\,e^{2x}\,dx$$
$$\int x\,e^{3x}\,dx$$
$$\int x\,e^{4x}\,dx$$

*Out[1]=* $E^x\,(-1+x)$

*Out[2]=* $E^{2x}\left(-\frac{1}{4}+\frac{x}{2}\right)$

*Out[3]=* $E^{3x}\left(-\frac{1}{9}+\frac{x}{3}\right)$

*Out[4]=* $E^{4x}\left(-\frac{1}{16}+\frac{x}{4}\right)$

Here is the general result found with *Mathematica*.

*In[5]:=*
$$\int x\,e^{nx}\,dx$$

*Out[5]=* $E^{nx}\left(-\frac{1}{n^2}+\frac{x}{n}\right)$

## ■ Section 7.7 Improper Integrals

Here are a couple examples of improper integrals computed directly with *Mathematica*.

In[6]:= $\int_1^{\infty} x\, e^{-x}\, dx$

Out[6]= $\frac{2}{E}$

In[7]:= $\int_0^2 \frac{1}{\sqrt{4 - x^2}}\, dx$

Out[7]= $\frac{\pi}{2}$

# CAS Exercise Examples for Chapter 8: Infinite Series

## ■ Section 8.1 Limits of Sequences of Numbers

*Mathematica* is a great tool for exploring sequences as the following example illustrates.

**Example:** Suppose $a_n = 2\arctan(n^3)$.

(a) Calculate and then plot the first 35 terms of the sequence. Determine the limit $L$ of the sequence.

(b) Find an integer N such that $|a_n - L| \le 0.0001$.

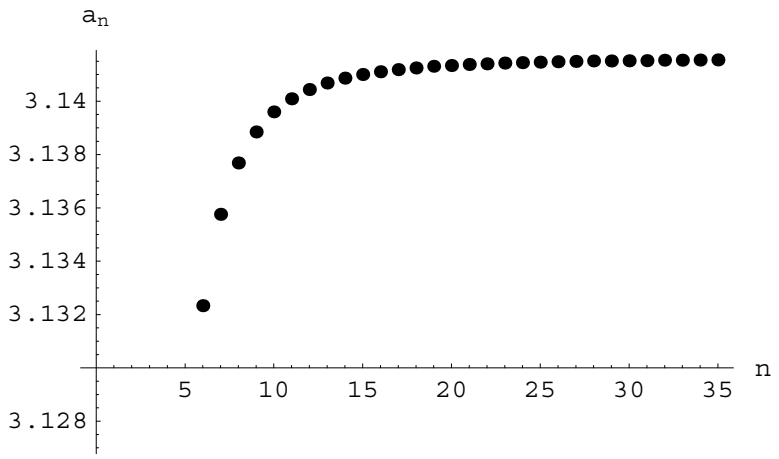
**Part (a)** The sequence is defined and then the `Table` command is used to display the first 35 terms in the sequence.

```
In[1]:= a[n_] = 2 ArcTan[n^3];
        seq = Table[N[a[n], 10], {n, 1, 35}]

Out[1]= {1.570796327, 2.892882664, 3.067552422, 3.110345196, 3.125592995, 3.13233346,
 3.135761766, 3.137686409, 3.138849171, 3.139592654, 3.140090024, 3.140435246,
 3.140682321, 3.140863791, 3.141000061, 3.141104372, 3.14118557, 3.141249718,
 3.141301066, 3.141342654, 3.141376694, 3.141404825, 3.141428275, 3.141447978,
 3.141464654, 3.141478862, 3.141491043, 3.141501546, 3.141510649, 3.14151858,
 3.141525519, 3.141531618, 3.141537001, 3.141541768, 3.141546006}
```

It appears that the sequence converges to $L = \pi$.

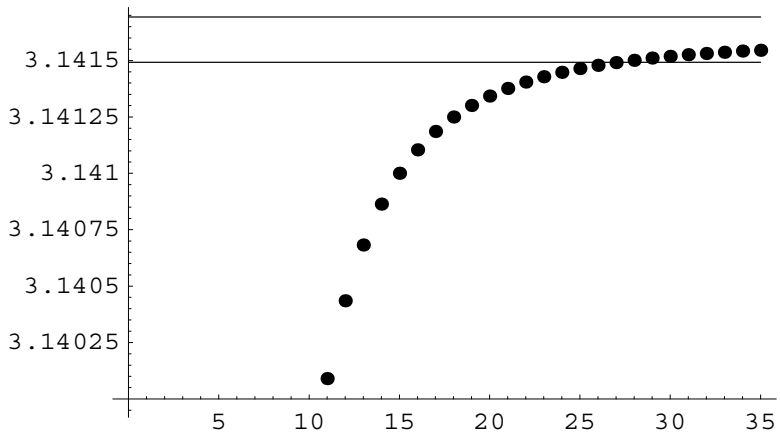
Next, the `ListPlot` command is used to plot the sequence. The option `PlotRange→All` is added to make sure all the points appear in the plot of the sequence. The option `AxesLabel -> {n, a_n}` is added to give appropriate labels to the horizontal and vertical axes.

```
In[2]:= seqplot = ListPlot[seq, PlotStyle -> PointSize[.02], AxesLabel -> {"n", "a_n"}];
```



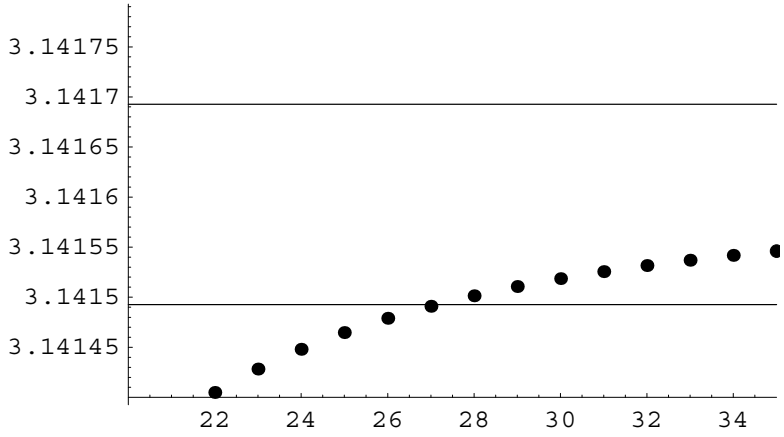

**Part (b)** To determine $n$ such that $|a_n - \pi| \le 0.0001$, the horizontal lines $y = \pi \pm 0.0001$ are plotted first.

```
In[3]:= band = Plot[{π - .0001, π + .0001}, {n, 0, 35}, DisplayFunction -> Identity];
        Show[{band, seqplot}, DisplayFunction -> $DisplayFunction];
```




The sequences begins to lie inside the lines $y = \pi \pm .0001$ somewhere between 25 and 30. Next, the graph is magnified to get a better view.

```
In[4]:= Show[{band, seqplot}, PlotRange -> {{20, 35}, {π - .0002, π + .0002}},
          DisplayFunction -> $DisplayFunction];
```




It is now clear that $|a_n - \pi| \le 0.0001$ for $n \ge 28$. To verify this, the value $|a_n - \pi|$ is computed for $n = 27$ and $n = 28$.

```
In[5]:= Abs[a[27] - π] // N
        Abs[a[28] - π] // N
Out[5]= 0.000101611
Out[6]= 0.0000911079
```

Therefore it appears that $N = 28$.

■

## Section 8.2 Subsequences, Bounded Sequences, and Picard's Method

**Example:** Let $a_1 = 1$ and $a_{n+1} = \frac{1}{1+a_n}$.

(a) Use *Mathematica* to calculate and plot the first 25 terms of the sequence. Does the sequence appear to converge and if so, to what value $L$?

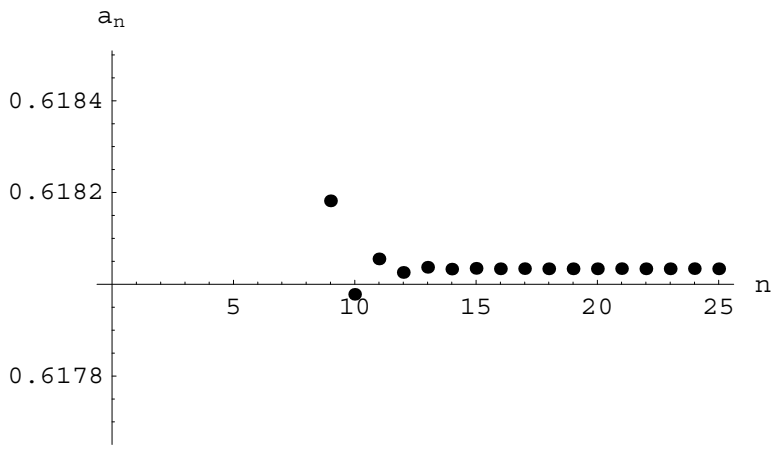
(b) If the sequences does converge, find an integer $N$ such that $|a_n - L| \le 0.00001$ for $n \ge N$.

**Part (a)** The following input cell defines $a_n$ recursively and then uses the `Table` command to display the first 25 terms of the sequence.

```
In[7]:= Clear[a];
        a[1] = 1;
        a[n_] := a[n] = 1/(1 + a[n - 1]);
        seq = Table[N[a[n], 10], {n, 1, 25}]
Out[7]= {1., 0.5, 0.6666666667, 0.6, 0.625, 0.6153846154, 0.619047619,
         0.6176470588, 0.6181818182, 0.6179775281, 0.6180555556,
         0.6180257511, 0.6180371353, 0.6180327869, 0.6180344478,
         0.6180338134, 0.6180340557, 0.6180339632, 0.6180339985, 0.618033985,
         0.6180339902, 0.6180339882, 0.618033989, 0.6180339887, 0.6180339888}
```

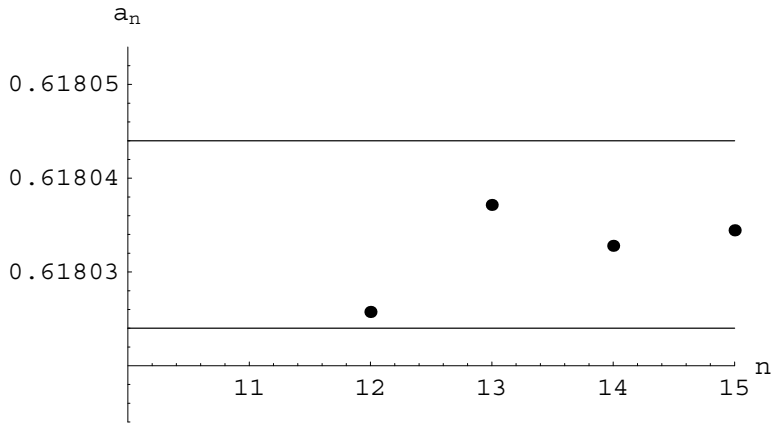
Now the `Plot` command is used to obtain a graph of the sequence.

```
In[8]:= seqplot = ListPlot[seq, PlotStyle -> PointSize[.02], AxesLabel -> {"n", "a_n"}];
```




**Part (b)** From part (a), it appears that $L \approx 0.618033989$. The value of $N$ needed so that $|a_n - L| \leq 0.00001$ for $n \geq N$ is determined by plotting the horizontal lines $y = L \pm 0.00001$ with the sequence.

```
In[9]:= L = 0.618033989;
        band = Plot[{L - .00001, L + .00001}, {x, 0, 25}, DisplayFunction -> Identity];
        Show[{seqplot, band}, DisplayFunction -> $DisplayFunction,
          PlotRange -> {{10, 15}, {L - .00002, L + .00002}}];
```




It appears that $N = 12$. To verify this, the value $|a_n - L|$ is computed for $n = 12$.

```
In[10]:= Abs[a[12] - L]
Out[10]= 8.23793 × 10^-6
```

## Section 8.4 Series of Nonnegative Terms

### ■ Computing Infinite Series with *Mathematica*

There are several ways of attempting to find the sum of an infinite series with *Mathematica*.

The first method is to define the nth partial sum of the series. For example, to find the value of $\sum_{n=1}^{\infty} \frac{1}{n(n+1)}$, the following nth partial sum can be defined.

```
In[11]:= s[n_] := Sum[1/(i (i + 1)), {i, 1, n}]
```

Now the value of $\lim_{n\to\infty} s_n$ is computed.

```
In[12]:= Limit[s[n], n → ∞]
Out[12]= 1
```

However, if this same method is applied to $\sum_{n=1}^{\infty} \frac{1}{n^2}$, the limit is not found.

In[13]:=
```
Clear[s];
s[n_] := Sum[1/i^2, {i, 1, n}];
Limit[s[n], n → ∞]
```

Out[13]=
$$\text{Limit}\left[\frac{\pi^2}{6} - \text{PolyGamma}[1, 1 + n], n \to \infty\right]$$

Another way of evaluating an infinite series to enter $\sum_{n=1}^{\infty} a_n$. Here are a couple of examples.

In[14]:=
```
Sum[1/n^2, {n, 1, ∞}]
```

Out[14]=
$$\frac{\pi^2}{6}$$

In[15]:=
```
Sum[1/(n^2 + n - 1), {n, 1, ∞}]
```

Out[15]=
$$\frac{4\,\text{Sec}\left[\frac{\sqrt{5}\,\pi}{2}\right]\left(\sqrt{5}\,\text{Cos}\left[\frac{\sqrt{5}\,\pi}{2}\right] + \pi\,\text{Sin}\left[\frac{\sqrt{5}\,\pi}{2}\right]\right)}{\sqrt{5}\left(-1 + \sqrt{5}\right)\left(1 + \sqrt{5}\right)}$$

Use the `N` command to see the numerical approximation of the last output. The value of the last output is represented by `%`.

In[16]:=
```
N[%]
```

Out[16]=
```
1.54625
```

Sometimes *Mathematica* will produce the output of an infinite series in terms of a nonelementary function as seen in the following cells.

In[17]:=
```
Sum[1/(n^3 + 1), {n, 1, ∞}]
```

Out[17]=
$$\frac{1 - \text{EulerGamma}}{\left(-1 + (-1)^{1/3}\right)\left(1 + (-1)^{1/3}\right)^2} + \frac{1}{3}\left(\text{PolyGamma}[0, 1 - (-1)^{1/3}] + (-1)^{2/3}\,\text{PolyGamma}[0, 1 - (-1)^{1/3}] - (-1)^{2/3}\,\text{PolyGamma}[0, 1 + (-1)^{2/3}]\right)$$

In this case, either execute the command `N[%]` or use `NSum` command as shown next.

In[18]:=
```
NSum[1/(n^3 + 1), {n, 1, ∞}]
```

Out[18]=
```
0.686503
```

## ■ Section 8.7 Taylor and Maclaurin Series

### Finding Taylor Polynomials with *Mathematica*

The *Mathematica* command `Series[f[x], {x, a, n}]` is used to produce the Taylor series generated by $f$ at $x = a$. Terms up to a power of $(x - a)^n$ will be displayed. Here is an example.

```
In[19]:= Series[x/(1 - x), {x, 2, 3}]
Out[19]= -2 + (x - 2) - (x - 2)^2 + (x - 2)^3 + O[x - 2]^4
```

The notation `O[x - 1]^4` represents terms of $(x - 1)$ to powers of 4 and higher, which are not explicitly displayed.

The *Mathematica* command `Normal[Series[f[x],{x,a, n}]]` will produce the Taylor polynomial of order $n$, $P_n(x)$, generated by $f$ at $x = a$.

```
In[20]:= Normal[Series[x/(1 - x), {x, 2, 3}]]
Out[20]= -4 - (-2 + x)^2 + (-2 + x)^3 + x
```

## ■ Cubic Approximations

Example: Let $f(x) = \frac{x}{\sqrt{1+x^2}}$. Complete each of the following.

(a) Find the Taylor polynomial $P_3(x)$ at $x = 0$ and plot $P_3(x)$ together with $f(x)$ over the interval $[-1, 1]$.

(b) Calculate the fourth derivative $f^{(4)}(c)$ associated with the remainder term for $P_3(x)$. Plot the derivative as a function of $c$ over the interval $[-1, 1]$ and estimate the maximum absolute value $M$.

(c) Calculate the remainder $R_3(x)$ over the interval $[-1, 1]$, using the value of $M$ found in part (b) in place of $f^{(4)}(c)$. Plot $R_3(x)$ over the interval $[-1, 1]$. Use the plot to determine the values of $x$ for which the function $f(x)$ can be replaced by $P_3(x)$ with an error of less than 0.01.

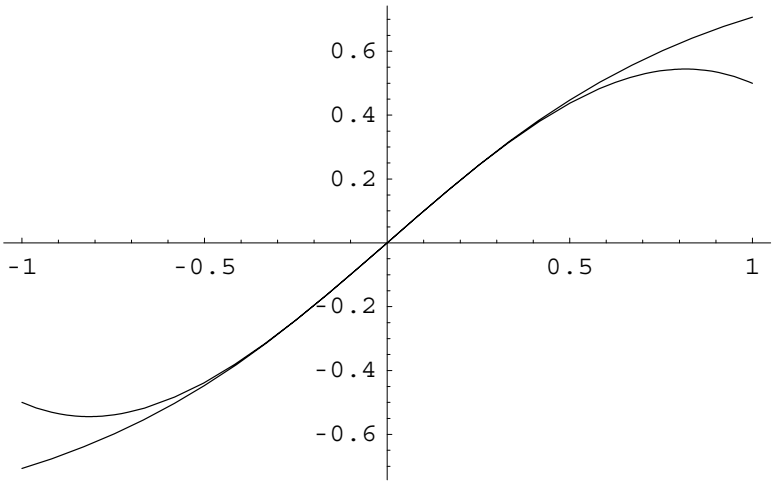
(d) Compare the estimated error $R_3(x)$ with the actual error $E_3(x) = |f(x) - P_3(x)|$ over the interval found in part (c).

**Part (a)** As previously described, the `Normal` and `Series` commands can combined to obtain the Taylor polynomial.

```
In[21]:= f[x_] = x/Sqrt[1 + x^2];
         p3[x_] = Normal[Series[f[x], {x, 0, 3}]]
Out[21]= x - x^3/2
```

The function and the Taylor polynomial can now be plotted.

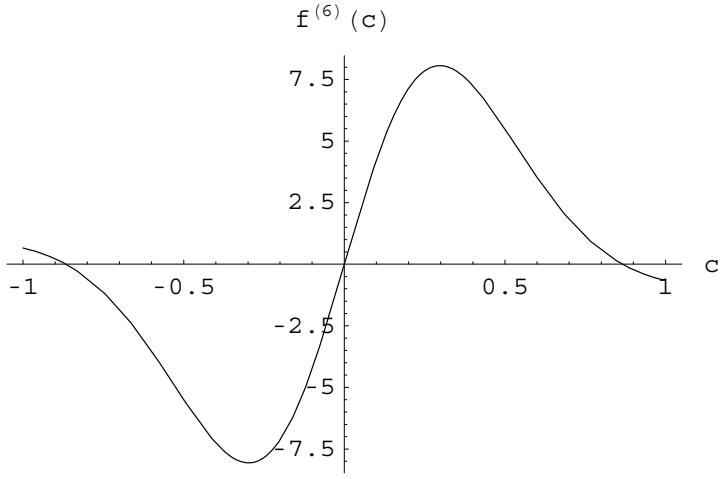
```
In[22]:= Plot[{f[x], p3[x]}, {x, -1, 1}];
```




**Part (b)** The *Mathematica* command `D[f[x],{x,n}]` will compute $f^{(n)}(x)$. After computing the fourth derivative,, a graph is obtained using the `Plot` command.

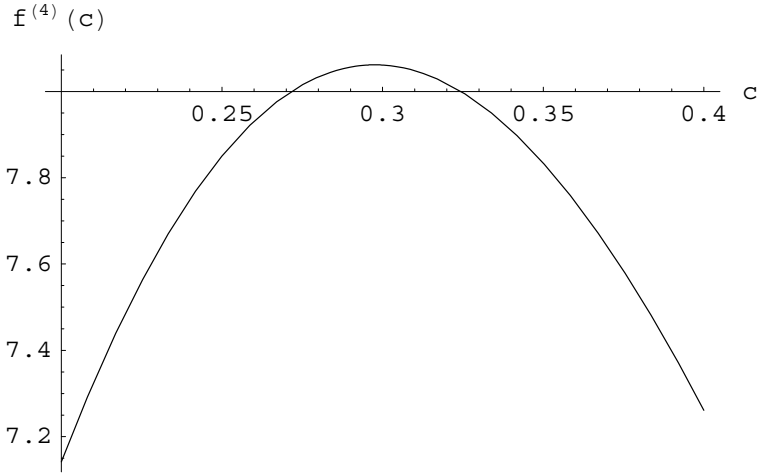
```
In[23]:= g[x_] = D[f[x], {x, 4}] // Simplify
         Plot[g[c], {c, -1, 1}, AxesLabel -> {c, "f(6)(c)"}, PlotRange -> All];
```

$$\text{Out[23]=}\quad -\frac{15\,x\,(-3+4\,x^2)}{(1+x^2)^{9/2}}$$



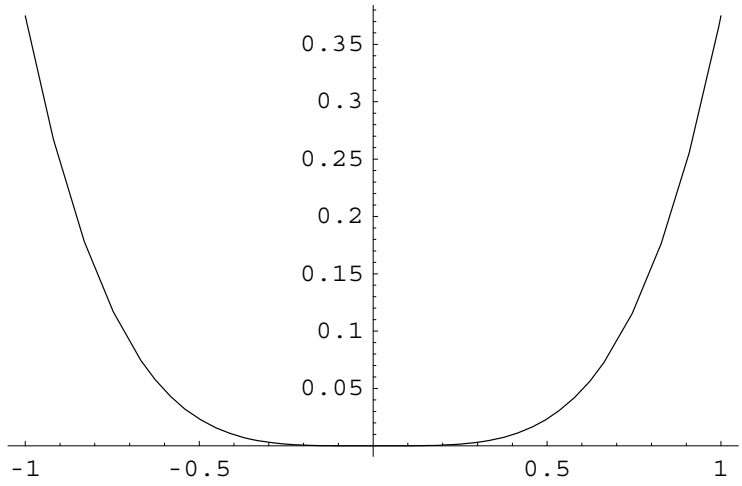

The maximum value of $|\,f^{(4)}(c)\,|$ is obtained near $x = \pm 0.3$. Zooming in on the graph reveals that $M = 9$ is a reasonable estimate of $M$.

```
In[25]:= Plot[g[c], {c, .2, .4}, AxesLabel -> {c, "f^(4)(c)"}, PlotRange -> All];
```



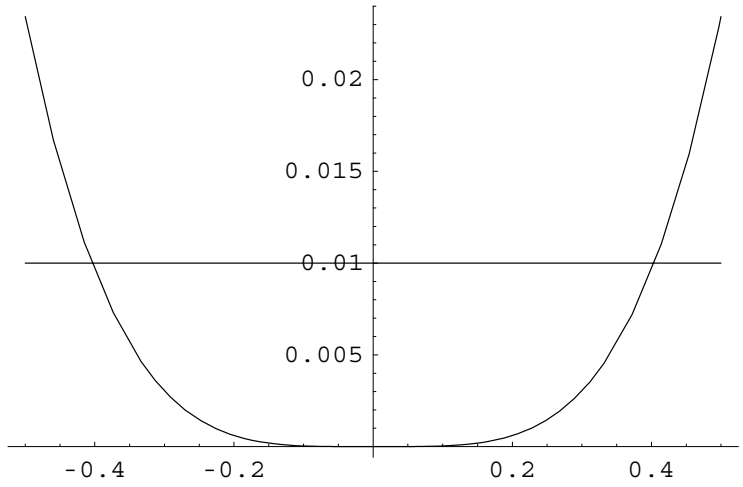

**Part (c)**: An approximation of $R_3(x)$ can now be formed and plotted.

```
In[26]:= M = 9;
         R3[x_] = M x^4 / 4!;
         Plot[R3[x], {x, -1, 1}, PlotRange -> All];
```

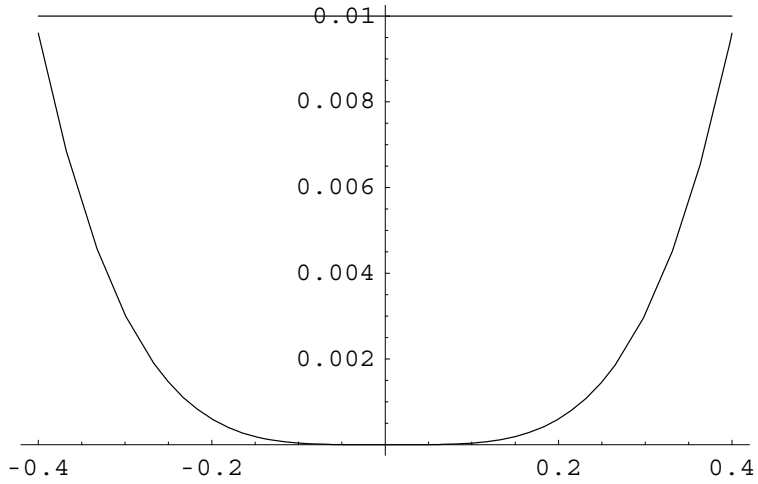



The goal now is to zoom in on the graph by trial and error to the point where $|R_3(x)| \le 0.01$ for all values of $x$ in the graph.

In[27]:= **Plot[{0.01, R$_3$[x]}, {x, -.5, .5}, PlotRange -> All];**



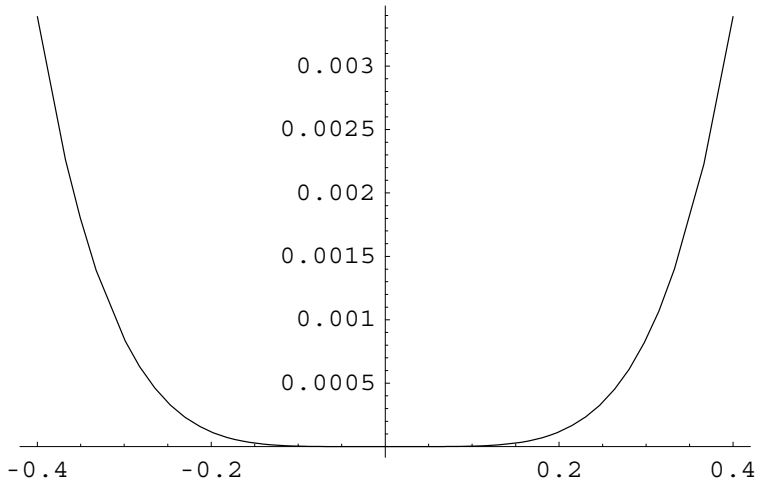

In[28]:= **Plot[{0.01, R$_3$[x]}, {x, -.4, .4}, PlotRange -> All];**




From the last graph just obtained, it is clear that $|R_3(x)| \leq 0.01$on $[-0.4, 0.4]$ implying that $|f(x) - P_3(x)| \leq 0.01$ on this interval.

**Part (d)** As the following output shows, the actual error $|f(x) - P_3(x)|$ is much smaller than 0.01 on [-0.4,0.4].

In[29]:= **Plot[Abs[f[x] - p$_3$[x]], {x, -.4, .4}, PlotRange -> All];**



# CAS Exercise Examples for Chapter 9: Vectors in the Plane and Polar Functions

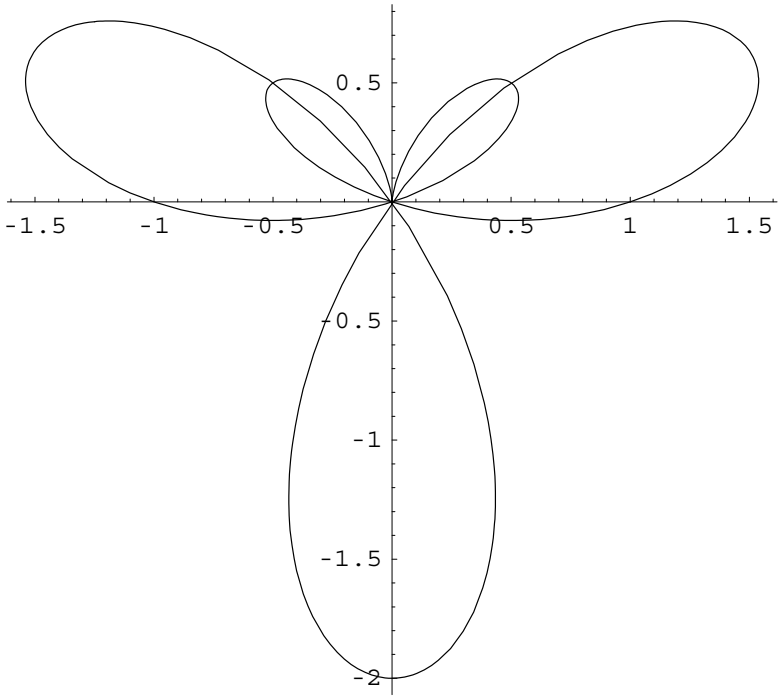
## ■ Section 9.5 Polar Coordinates and Graphs

### Polar Graphing

The *Mathematica* command `PolarPlot[r[θ],{θ,θmin,θmax}]` will plot the graph of a polar function. But before using this command, it must be loaded into memory first using the `<< Graphics`Graphics`` command. Study the following example.

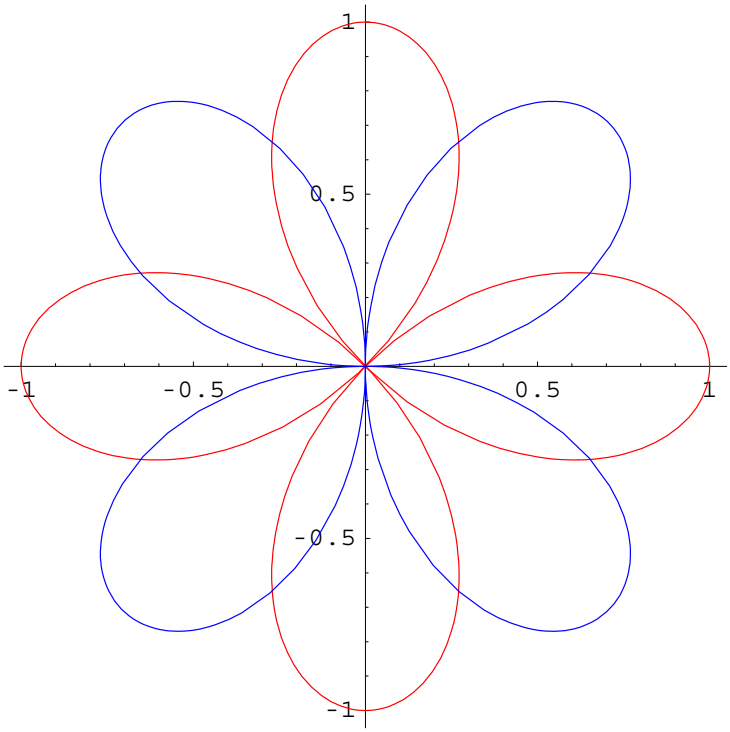
*In[1]:=*
```
<< Graphics`Graphics`
```

*In[2]:=*
```
PolarPlot[Sin[3 θ] + Cos[2 θ], {θ, 0, 2 π}];
```



The following is an example of plotting two functions simultaneously in different colors in a way completely analogous to the `Plot` command. The option `PlotStyle→{RGBColor[1,0,0],RGBColor[0,0,1]}` will plot the graph of $\cos(2\theta)$ in red and the graph of $\sin(2\theta)$ in blue.

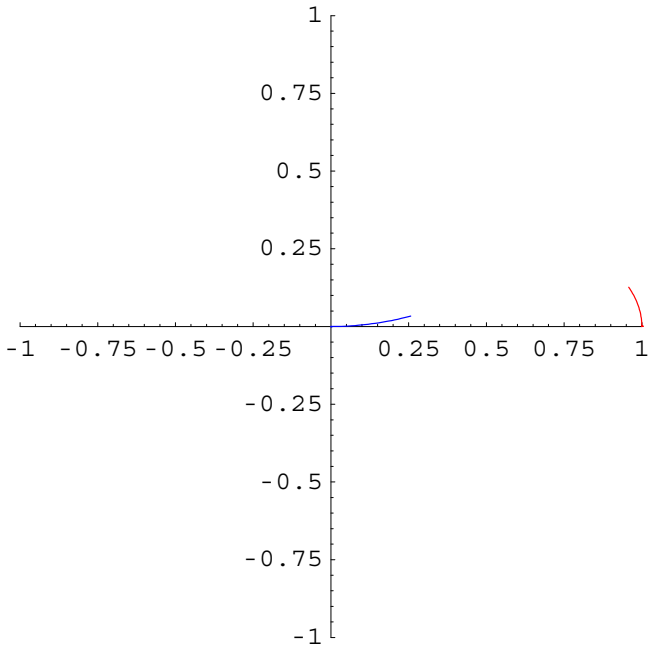
```
In[3]:= PolarPlot[{Cos[2 θ], Sin[2 θ]}, {θ, 0, 2 π},
          PlotStyle → {RGBColor[1, 0, 0], RGBColor[0, 0, 1]}];
```




### Finding Simultaneous Intersections

In the last part of Section 9.5 in the textbook, the simultaneous intersections of $r = \cos 2\theta$ and $r = \sin 2\theta$ are discussed. To visualize these intersections with *Mathematica*, the Table command can be used to plot frames of a movie showing the graphs being formed. Study the following input and output (only the first frame of the movie is shown in the hard copy of this manual). Double click on any frame to animate the frames. How many times do the graphs intersect simultaneously?

```
In[4]:= Table[PolarPlot[{Cos[2 θ], Sin[2 θ]},
          {θ, 0, θ0}, PlotStyle → {RGBColor[1, 0, 0], RGBColor[0, 0, 1]},
          PlotRange -> {{-1, 1}, {-1, 1}}], {θ0, π/24, 2 π, π/24}];
```


1
0.75
0.5
0.25
-1
-0.75
-0.5
-0.25
0.25
0.5
0.75
1
-0.25
-0.5
-0.75
-1

# CAS Exercise Examples for Chapter 10: Vectors and Motion in Space

---

## ■ Section 10.4 Cylinders & Quadric Surfaces

As indicated at the end of Section 10.4 in the textbook, quadric surfaces can be plotted using a CAS such as *Mathematica* after first describing the equation in parametric form. In the following examples, quadric surface equations of the form $\frac{x^2}{a^2} + \frac{y^2}{b^2} = f(z)$ are given where $f$ is a function of $z$. By letting $x = a\sqrt{f(z)}\cos(\theta)$ and $y = b\sqrt{f(z)}\sin(\theta)$, then $\frac{\left(a\sqrt{f(z)}\cos(\theta)\right)^2}{a^2} + \frac{\left(b\sqrt{f(z)}\sin(\theta)\right)^2}{b^2} = \frac{a^2 f(z)\cos^2(\theta)}{a^2} + \frac{b^2 f(z)\sin^2(\theta)}{b^2} = f(z)(\cos^2\theta + \sin^2\theta) = f(z)$

and therefore the equation $\frac{x^2}{a^2} + \frac{y^2}{b^2} = f(z)$ is satisfied. To display the graph of the quadric surface, use the *Mathematica* command

```
ParametricPlot3D[{a Sqrt[f[z]] Cos[θ], b Sqrt[f[z]] Sin[θ], z}, {θ, 0, 2 π}, {z, z_min, z_max}].
```

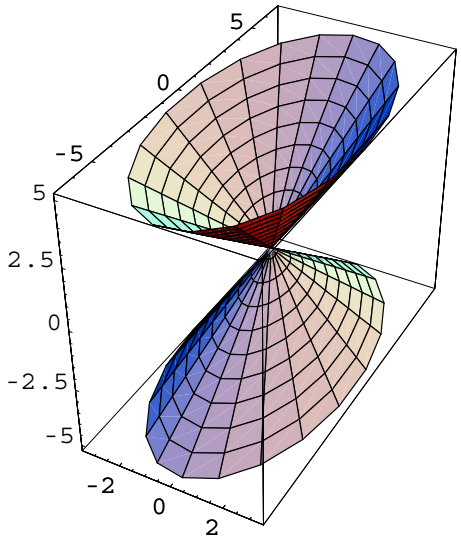
**Example:** Use *Mathematica* to plot the following surfaces.

(a) $\frac{x^2}{4} + \frac{y^2}{16} = \frac{z^2}{9}$

(b) $\frac{y^2}{25} - \frac{z^2}{16} = \frac{x^2}{9} + 1$

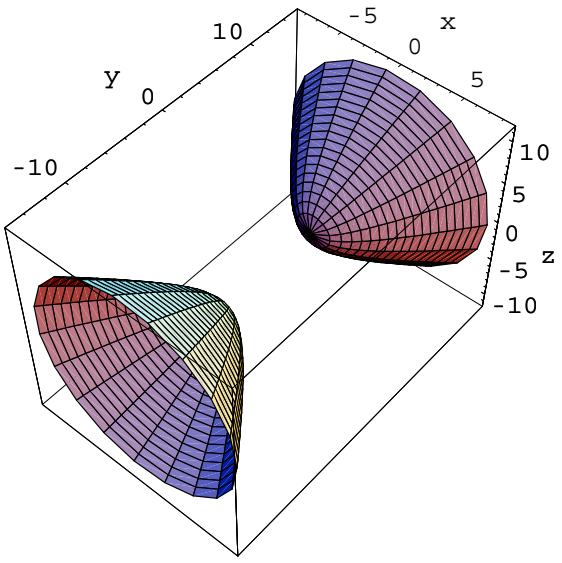
**Part (a)** In this case, $a^2 = 4$, $b^2 = 16$ and $f(z) = \frac{z^2}{9}$. Therefore, let $x = 2\sqrt{\frac{z^2}{9}}\cos(\theta) = \frac{2z}{3}\cos(\theta)$ and let $y = 4\sqrt{\frac{z^2}{9}}\sin(\theta) = \frac{4z}{3}\sin(\theta)$.

```
In[1]:= ParametricPlot3D[{2 z/3 Cos[θ], 4 z/3 Sin[θ], z}, {θ, 0, 2 π}, {z, -5, 5}];
```




**Part (b)** Rearrange the equation $\frac{y^2}{25} - \frac{z^2}{16} = \frac{x^2}{9} + 1$ to get $\frac{x^2}{9} + \frac{z^2}{16} = \frac{y^2}{25} - 1$. This is a hyperboloid of two sheets (why?). In this case, the equation is satisfied if $x = 3\cos(\theta)\sqrt{\frac{y^2}{25} - 1}$, $z = 4\sin(\theta)\sqrt{\frac{y^2}{25} - 1}$. Notice that $x$ and $z$ are defined only when $y \le -5$ or $y \ge 5$ which corresponds to the two parts of the graph. Study the following input commands. The option `BoxRatios→{1,1.5,1}` scales the plot so that the length in the y direction is 1.5 times as long as the lengths in the other two directions.

```
In[2]:= backpart = ParametricPlot3D[{3 Cos[θ] Sqrt[y^2/25 - 1], y, 4 Sin[θ] Sqrt[y^2/25 - 1]},
          {θ, 0, 2 π}, {y, 5, 15}, DisplayFunction -> Identity];
        frontpart = ParametricPlot3D[{3 Cos[θ] Sqrt[y^2/25 - 1], y, 4 Sin[θ] Sqrt[y^2/25 - 1]},
          {θ, 0, 2 π}, {y, -15, -5}, DisplayFunction -> Identity];
        Show[{frontpart, backpart}, DisplayFunction -> $DisplayFunction, BoxRatios ->
          {1, 1.5, 1}, ViewPoint -> {1.442, -1.680, 2.559}, AxesLabel -> {x, y, z}];
```




## ■ Section 10.5 Vector-Valued Functions and Space Curves

Here is an example similar to CAS exercises found in Section 10.5.

**Example:** Let $\mathbf{r}(t) = [(4 + \sin(10t))\cos(t)]\mathbf{i} + [(4 + \sin(10t))\cos(t)]\,\mathbf{j} + (2\cos(10t))\mathbf{k}$ and complete each of the following.

(a) Plot the space curve traced out by the position vector for $0 \le t \le 2\pi$.

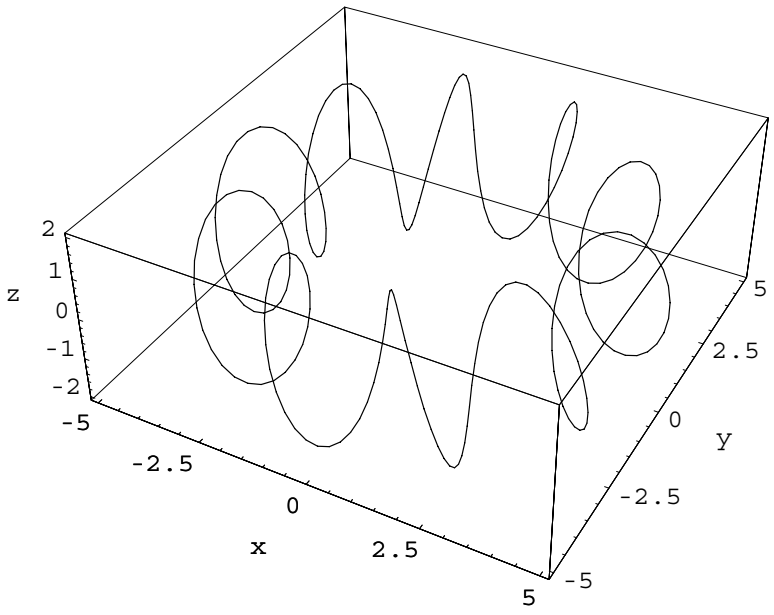
(b) Find the components of the velocity vector $dr/dt$.

(c) Evaluate the velocity vector at $t = \frac{\pi}{6}$ and determine the equation of the tangent line to the curve at $\mathbf{r}(\frac{\pi}{6})$.

(d) Plot the tangent line together with the curve.

**Part (a)** First, the $x$, $y$ and $z$ components of the curve are defined parametrically and then the curve is plotted.

```
In[3]:= x[t_] = (4 + Sin[10 t]) Cos[t];
        y[t_] = (4 + Sin[10 t]) Sin[t];
        z[t_] = 2 Cos[10 t];
        spacecurve = ParametricPlot3D[{x[t], y[t], z[t]},
           {t, 0, 2 π}, PlotPoints -> 250, AxesLabel -> {x, y, z}];
```




**Part (b):** Now the derivative of each component is computed to obtain the velocity vector.

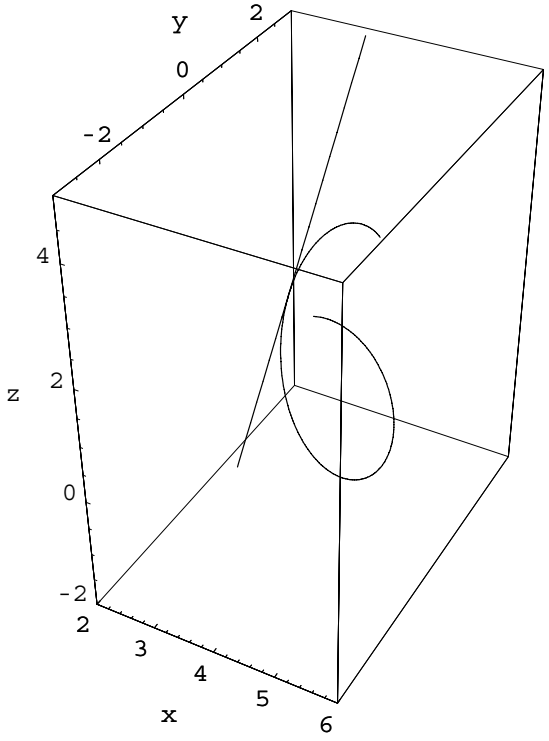
```
In[4]:= v[t_] = {x'[t], y'[t], z'[t]}
Out[4]= {10 Cos[t] Cos[10 t] - Sin[t] (4 + Sin[10 t]),
         10 Cos[10 t] Sin[t] + Cos[t] (4 + Sin[10 t]), -20 Sin[10 t]}
```

**Part (c)** Next, the velocity vector is evaluated and the parametric equations of the tangent line are computed.

```
In[5]:= v[π/6] // N
Out[5]= {2.76314, 5.2141, 17.3205}

In[6]:= xl[t_] := x[π/6] + x'[π/6] t;
        yl[t_] := y[π/6] + y'[π/6] t;
        zl[t_] := z[π/6] + z'[π/6] t;
```

**Part (d)** The space curve and the tangent line are displayed in the following output cell. Instead of plotting the curve for values of $t$ from 0 to $2\pi$, a smaller interval of values is used so that the tangent line is easier to see.

In[7]:=
```
tanline = ParametricPlot3D[{xl[t], yl[t], zl[t]},
   {t, -π/4, π/4}, DisplayFunction -> Identity];
spacecurve = ParametricPlot3D[{x[t], y[t], z[t]}, {t, 0, π/4},
   PlotPoints -> 250, AxesLabel -> {x, y, z}, DisplayFunction -> Identity];
Show[{spacecurve, tanline}, PlotRange -> {{2, 6}, {-3, 3}, {-2, 5}},
  DisplayFunction -> $DisplayFunction];
```



## Section 10.7 The TNB Frame; Tangential and Normal Components of Acceleration

**Example:** Let $\mathbf{r}(t) = (\cos t)\mathbf{i} + (2 - \frac{1}{2}\sin t)\mathbf{j}$, $0 \le t \le 2\pi$ and let $t_0 = \frac{\pi}{4}$. Complete each of the following steps.

(a) Plot the plane curve over the specified interval.

(b) Calculate the curvature $\kappa$ of the curve at the given point $t_0$.

(c) Find the unit normal vector $\mathbf{N}$ at $t_0$.

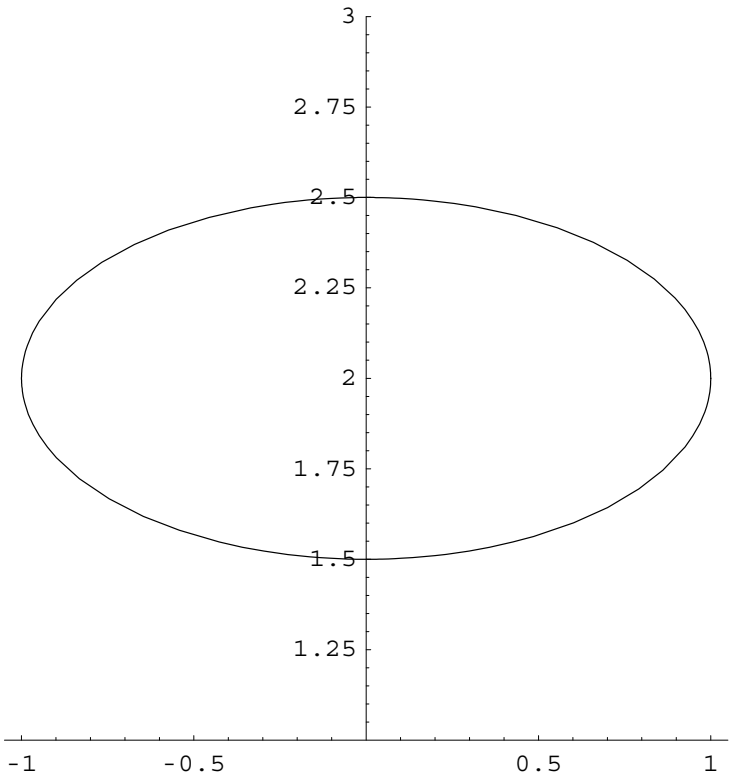
(d) If $\mathbf{C} = a\mathbf{i}+b\mathbf{j}$ is the vector from the origin to the center $(a, b)$ of the osculating circle, find the center $\mathbf{C}$ from the vector equation

$\mathbf{C}=\mathbf{r}(t_0) + \frac{1}{\kappa(t_0)}\mathbf{N}(t_0)$. The point $P(x_0, y_0)$ on the curve is given by the position vector $\mathbf{r}(t_0)$.

(e) Plot the curve and the osculating circle together.

**Part (a)** The curve is plotted parametrically with *Mathematica*.

```
In[8]:= x[t_] = Cos[t]; y[t_] = 2 - 1/2 Sin[t];
        curve =
          ParametricPlot[{x[t], y[t]}, {t, 0, 2 π}, AspectRatio -> 1, PlotRange -> {1, 3}];
```




**Part (b)** Using the formula found in Exercise 24 in your text, the value of $\kappa$ is computed.

```
In[9]:= κ[t_] = Abs[x'[t] y''[t] - y'[t] x''[t]] / (x'[t]^2 + y'[t]^2)^(3/2);
         κ[π/4] // N
Out[9]= 1.01193
```

**Part (c)** Since `N` is a reserved *Mathematica* word, you can instead use `n` to represent the unit normal vector. See Exercise 25 to see why the following formula was used to compute the normal vector. Is this the formula you should always use to find the unit normal vector? Why or why not?

```
In[10]:= n[t_] = {y'[t], -x'[t]} / Sqrt[x'[t]^2 + y'[t]^2];
         n[π/4]
Out[10]= {-1/Sqrt[5], 2/Sqrt[5]}
```

**Part (d)** The center of the osculating circle is found using the following input cell.

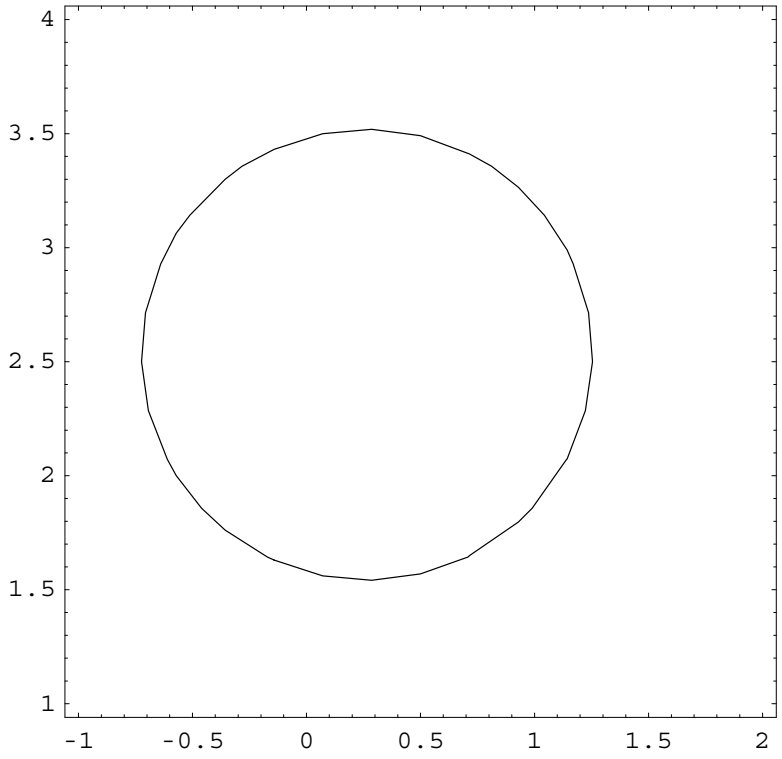
```
In[11]:= r[t_] = {x[t], y[t]};
         c = r[π/4] + 1/κ[π/4] n[π/4] // N
Out[11]= {0.265165, 2.53033}
```

**Part (e)** The *Mathematica* command

```
ContourPlot[f[x,y],{x, xmin, xmax},{y, ymin,  ymax},ContourShading->False, Contours->{c}]
```

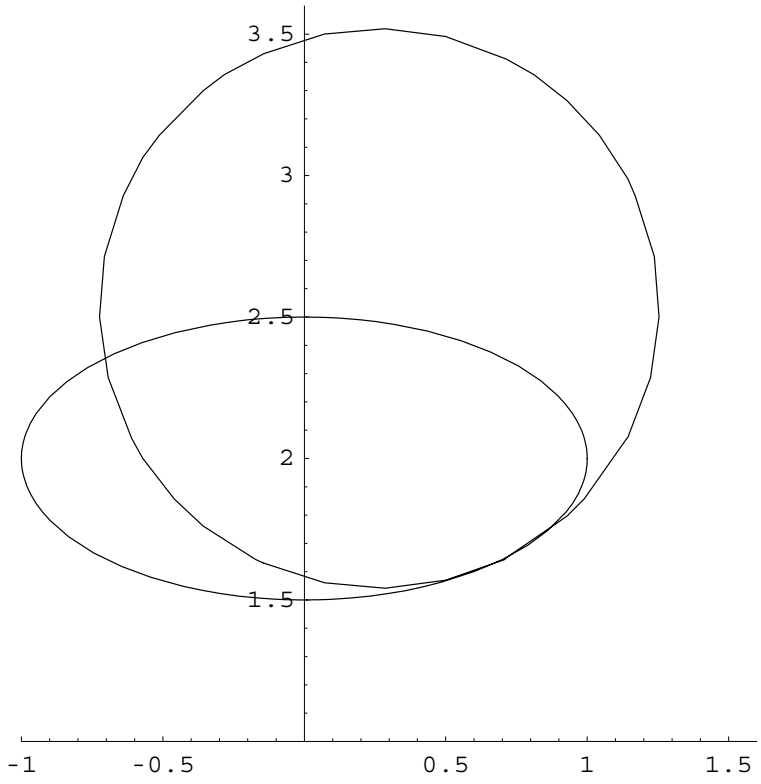
will plot the graph of $f(x, y) = c$. Therefore the following command is used to graph the osculating circle.

In[12]:=
```
osccir = ContourPlot[(x - c[[1]])^2 + (y - c[[2]])^2, {x, -1, 2},
  {y, 1, 4}, ContourShading -> False, Contours -> {1/κ[π/4]^2}];
```




Now the curve and circle are shown together.

In[13]:= **Show[{curve, osccir}, PlotRange -> {{-1, 1.6}, {1, 3.6}}, AspectRatio -> 1];**



# CAS Exercise Examples for Chapter 11: Multivariable Functions and Their Derivatives

---

## ■ Section 11.1 Functions of Several Variables

### Explicit Surfaces

In this section, you are asked to plot surfaces and level curves. The first example below uses the command

```
Plot3D[f[x,y],{x,xmin,xmax},{y,ymin,ymax}]
```

to plot the surface of $f(x, y)$ over the rectangle xmin $\le x \le$ xmax, ymin $\le y \le$ ymax. The *Mathematica* command

```
ContourPlot[f[x,y],{x,xmin,xmax},{y,ymin,ymax},ContourShading →False]
```

will plot level curves of $f(x, y)$ in the rectangle xmin $\le x \le$ xmax, ymin $\le y \le$ ymax. Deleting the option `ContourShading→False` will result in the level curves being shaded, with lighter shading corresponding to higher parts of the surface of $f(x, y)$. Adding the option `Contours` $\rightarrow$ `{`$c_1$`, `$c_2$`, ... `$c_n$`}` will instruct *Mathematica* to plot the level curves corresponding to $f(x, y) = c_i$ for $i = 1, 2, \ldots, n$.

**Example:** Consider the function $f(x, y) = e^{-x^2-y^2}$ over the rectangle $-2 \le x \le 2$, $-2 \le y \le 2$.

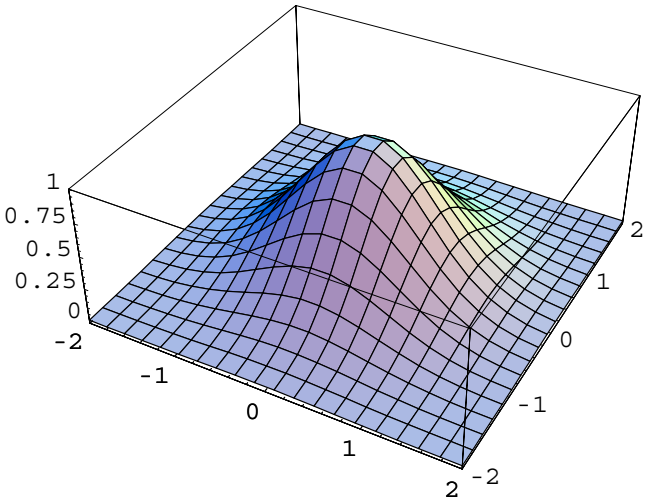
(a) Plot the surface over the given rectangle.

(b) Plot several level curves in the rectangle.

(c) Plot the level curve of $f$ though the point (1, 1).

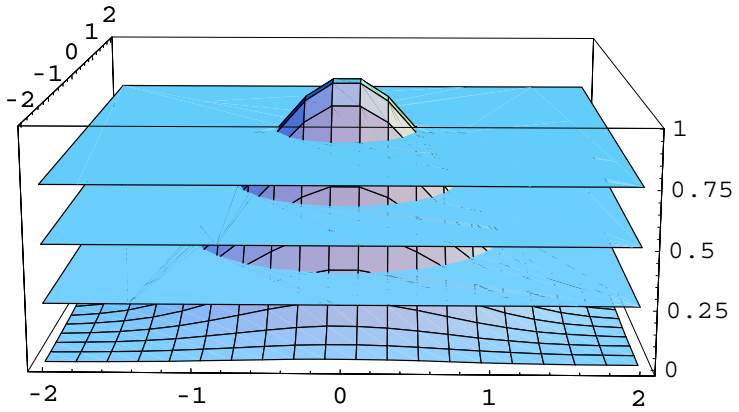
**Part (a)** The graph of the surface is obtained using the `Plot3D` command. The option `PlotPoints->20` increases the resolution of the displayed graph (the default value is `PlotPoints->15`).

```
In[1]:= f[x_, y_] = e^(-x^2-y^2);
        surface = Plot3D[f[x, y], {x, -2, 2}, {y, -2, 2}, PlotPoints -> 20];
```



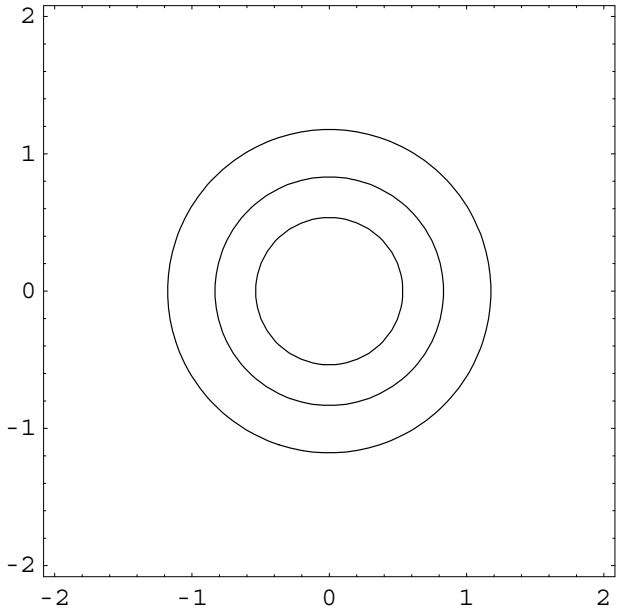

**Part (b)** Before plotting some level curves of the surface, you might find it helpful to see the three-dimensional images represented by level curves. For example, the following `Plot3D` commands are used to display the level curves corresponding to the horizontal planes $z = .25$, $z = .5$ and $z = .75$. The `ViewPoint` option (found under the Input menu), contained in the `Show` command, allows you to control the angle at which you view the image.

```
In[2]:= plane1 = Plot3D[.25, {x, -2, 2},
           {y, -2, 2}, PlotPoints -> 2, DisplayFunction -> Identity];
        plane2 = Plot3D[.5, {x, -2, 2}, {y, -2, 2}, PlotPoints -> 2,
           DisplayFunction -> Identity];
        plane3 = Plot3D[.75, {x, -2, 2}, {y, -2, 2}, PlotPoints -> 2,
           DisplayFunction -> Identity];
        Show[{surface, plane1, plane2, plane3}, DisplayFunction -> $DisplayFunction,
          ViewPoint -> {0.072, -3.303, 0.730}];
```



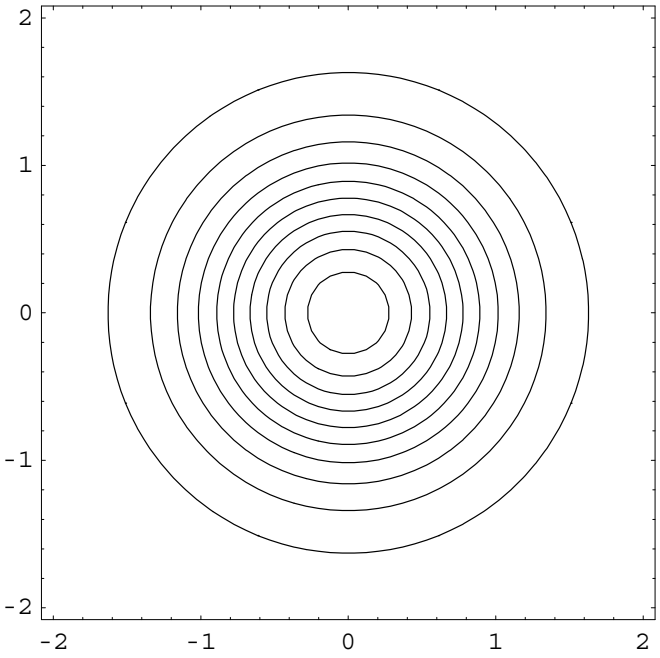

Now to show the actual level curves, the following `ContourPlot` command is executed.

```
In[3]:= ContourPlot[f[x, y], {x, -2, 2}, {y, -2, 2},
          PlotPoints -> 50, ContourShading → False, Contours → {.25, .5, .75}];
```



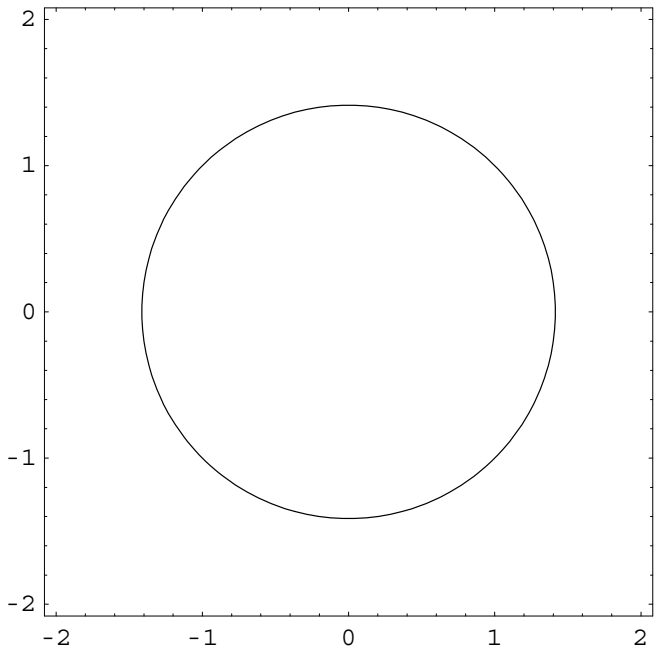

Deleting the `Contours` option allows *Mathematica* to choose the level curves to be plotted.

```
In[4]:= ContourPlot[f[x, y], {x, -2, 2},
          {y, -2, 2}, PlotPoints -> 50, ContourShading → False];
```




**Part (c)** The following graph plots the level curve passing through (1, 1). Why?

```
In[5]:= ContourPlot[f[x, y], {x, -2, 2}, {y, -2, 2},
          PlotPoints -> 50, ContourShading → False, Contours → {f[1, 1]}];
```




### Implicit Surfaces

The package **ContourPlot3D** contains the command

`ContourPlot3D[f[x,y,z],{x, xmin, xmax},{y, ymin,  ymax},{z,zmin,zmax},Contours->{c)]`

which will plot the level surface $f(x, y, z) = c$.

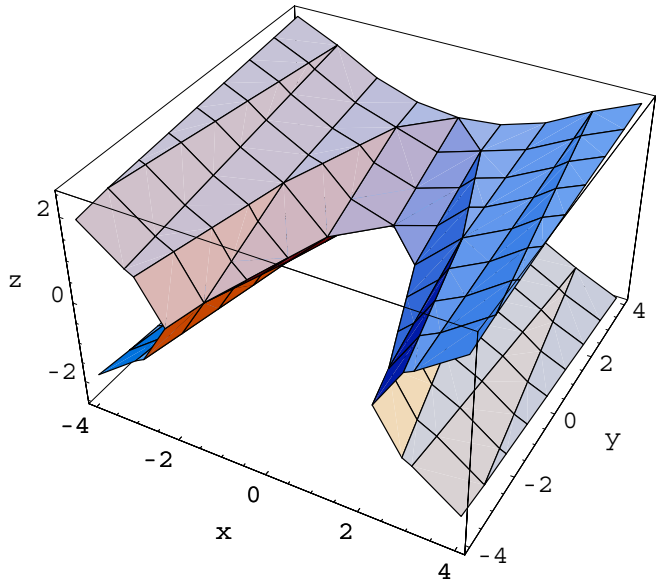
**Example:** Plot the level surface $x^2 + y - 3z^2 = 1$.

First, the package containing `ContourPlot3D` must be loaded.

```
In[6]:= << Graphics`ContourPlot3D`
```

Now the surface is plotted where the options `Axes->True` and `AxesLabel->{x,y,z}` are added so that the $x$, $y$ and $z$ axes are labeled.

In[7]:=
```
ContourPlot3D[x^2 + y - 3 z^2, {x, -4, 4}, {y, -4, 4},
  {z, -4, 4}, Contours -> {1}, Axes -> True, AxesLabel -> {x, y, z}];
```




### Parametrized Surfaces

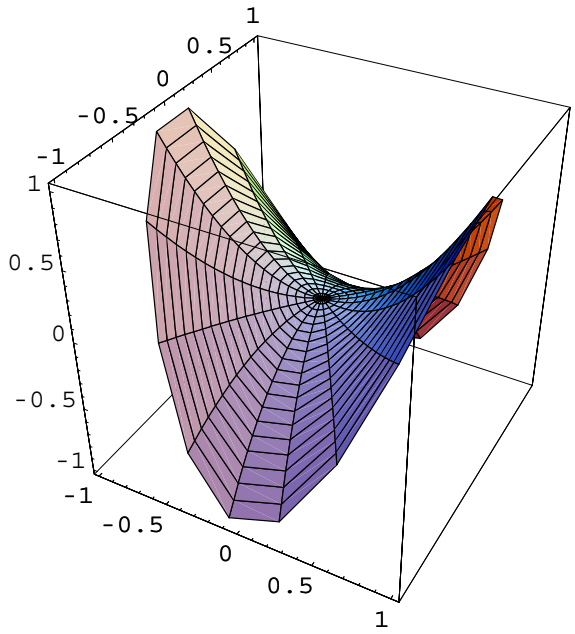
The *Mathematica* command

`ParametricPlot3D[{f[u, v], g[u, v], h[u, v]}, {u, umin, umax}, {v, umin, umax}]`

can be used to plot a surface described by the parametric equations

$x = f(u, v)$, $y = g(u, v)$ and $z = h(u, v)$. Consider the following example.

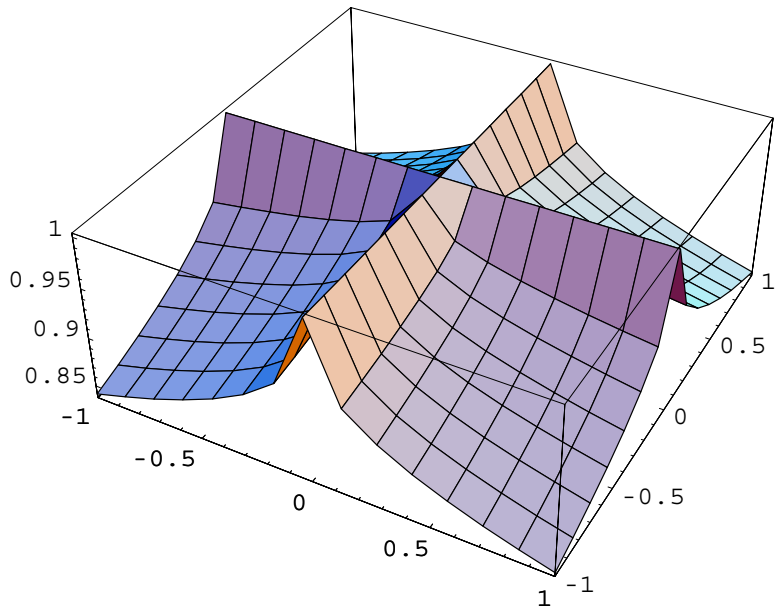
In[8]:=
```
ParametricPlot3D[
  {u Cos[v], u Sin[v], u^2 (Cos[v]^2 - Sin[v]^2)}, {u, 0, 1}, {v, 0, 2 π}];
```




## ■ Section 11.2 Limits and Continuity in Higher Dimensions

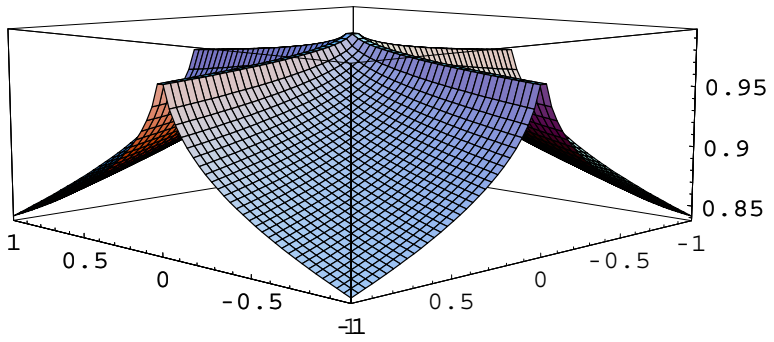
Using the Plot3D command, some interesting surfaces can be plotted such as the graph of $f(x, y) = \frac{\sin\left(\sqrt[5]{|x\,y|}\right)}{\sqrt[5]{|x\,y|}}$.

In[9]:=
```
Plot3D[Sin[(Abs[x y])^(1/5)] / (Abs[x y])^(1/5), {x, -1, 1}, {y, -1, 1}];
```



The `ViewPoint` option (under the Input menu) can be used to view a surface from a different angle.

```
In[10]:= Plot3D[Sin[Sqrt[5, Abs[x y]]] / Sqrt[5, Abs[x y]], {x, -1, 1}, {y, -1, 1},
           PlotPoints -> 60, ViewPoint -> {-2.377, 2.363, 0.464}];
```




## ■ Section 11.7 Extreme Values and Saddle Points

### Computing Partial Derivatives

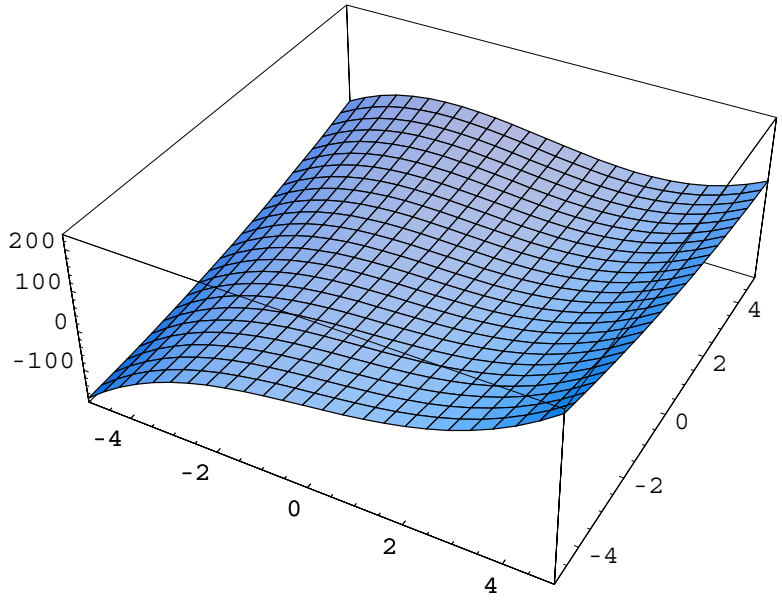
The command $\partial_x$ `f[x, y]` or `D[f[x,y],x]` can be used to compute $f_x(x, y)$ and the command $\partial_{x,y}$ `f[x, y]` or `D[f[x,y],x]` can be used to find $f_{xy}(x, y)$.

```
In[11]:= ∂x Sin[x^2 y]
Out[11]= 2 x y Cos[x^2 y]

In[12]:= ∂x,y Sin[x^2 y]
Out[12]= 2 x Cos[x^2 y] - 2 x^3 y Sin[x^2 y]

In[13]:= D[Sin[x^2 y], x, y]
Out[13]= 2 x Cos[x^2 y] - 2 x^3 y Sin[x^2 y]
```

**Example:** Consider the function $f(x, y) = x^3 + y^2 - 3xy$ on $-5 \le x \le 5,\ -5 \le y \le 5$. Plot the function over the given rectangle and then find and classify the critical points.

Here is a plot of the function.

```
In[14]:= Clear[f];
         f[x_, y_] = x^3 + y^2 - 3 x y ;
         Plot3D[f[x, y], {x, -5, 5}, {y, -5, 5}, PlotPoints -> 25];
```




Now the first partial derivatives are found. (Ignore the warning messages.)

```
In[15]:= xparder = ∂x f[x, y] // Simplify
         yparder = ∂y f[x, y] // Simplify
Out[15]= 3 (x^2 - y)
         General::spell1 : Possible spelling error: new symbol
            name "yparder" is similar  to existing symbol "xparder".
Out[16]= -3 x + 2 y
```

The command `Solve[{m[x,y]==0,n[x,y]==0},{x,y}]` can be used to simultaneously solve $m(x, y) = 0$ and $n(x, y) = 0$.

```
In[17]:= sol = Solve[{xparder == 0, yparder == 0}, {x, y}]
Out[17]= {{y → 0, x → 0}, {y → 9/4, x → 3/2}}
```

Now the second partial derivatives of the function are computed. (Ignore the

warning messages.)

```
In[18]:= xxparder = ∂x,x f[x, y] // Simplify;
         yyparder = ∂x,y f[x, y] // Simplify;
         xyparder = ∂x,y f[x, y] // Simplify;
         discrim = xxparder yyparder - xyparder^2;
         General::spell1 : Possible spelling error: new symbol
            name "xxparder" is similar  to existing symbol "xparder".
         General::spell1 : Possible spelling error: new symbol
            name "yyparder" is similar  to existing symbol "yparder".
         General::spell :
          Possible spelling error: new symbol name "xyparder" is similar
            to existing symbols {xparder, xxparder, yparder, yyparder}.
```

From the following work, you can conclude that both critical points are

saddle points.

```
In[19]:= xxparder /. sol[[1]]
         discrim /. sol[[1]]
Out[19]= 0
Out[20]= -9
In[21]:= xxparder /. sol[[2]]
         discrim /. sol[[2]]
Out[21]= 9
Out[22]= -36
```

## ■ Section 11.8 Lagrange Multipliers

**Example:** Minimize the function $f(x, y) = xz + xy$ subject to the constraints $x^2 + z^2 - 4 = 0$ and $x^2 + y^2 - 5 = 0$.

**Step one:** First the functions $f$, $g_1$, $g_2$ and $h$ are defined as shown in the following input cell.

```
In[23]:= Clear[f, h];
         f = x z + x y;
         g1 = x^2 + z^2 - 4;
         g2 = x^2 + y^2 - 5;
         h = f - λ1 g1 - λ2 g2;
```

**Step two:** The partial derivatives are then computed and placed in equations which all have zero on the right side.

In[24]:=
```
eq1 = ∂x h == 0;
eq2 = ∂y h == 0;
eq3 = ∂z h == 0;
eq4 = ∂λ1 h == 0;
eq5 = ∂λ2 h == 0;
```

**Step Three:** The `Solve` command is used to solve the system of equations formed in step two.

In[25]:=
```
sol = Solve[{eq1, eq2, eq3, eq4, eq5}, {x, y, z, λ1, λ2}]
```

Out[25]=
$$\left\{\left\{\lambda_1 \to -\frac{\sqrt{5}}{4},\ \lambda_2 \to -\frac{1}{\sqrt{5}},\ y \to -\frac{5}{3},\ z \to -\frac{4}{3},\ x \to \frac{2\sqrt{5}}{3}\right\},\right.$$
$$\left\{\lambda_1 \to -\frac{\sqrt{5}}{4},\ \lambda_2 \to -\frac{1}{\sqrt{5}},\ y \to \frac{5}{3},\ z \to \frac{4}{3},\ x \to -\frac{2\sqrt{5}}{3}\right\},$$
$$\left\{\lambda_1 \to \frac{\sqrt{5}}{4},\ \lambda_2 \to \frac{1}{\sqrt{5}},\ y \to -\frac{5}{3},\ z \to -\frac{4}{3},\ x \to -\frac{2\sqrt{5}}{3}\right\},$$
$$\left.\left\{\lambda_1 \to \frac{\sqrt{5}}{4},\ \lambda_2 \to \frac{1}{\sqrt{5}},\ y \to \frac{5}{3},\ z \to \frac{4}{3},\ x \to \frac{2\sqrt{5}}{3}\right\}\right\}$$

**Step Four:** The function $f$ is then evaluated at each solution to determine the extreme values subject to the constraints asked for in the exercise. What are the extreme values based upon the following work?

In[26]:=
```
f /. sol[[1]]
f /. sol[[2]]
f /. sol[[3]]
f /. sol[[4]]
```

Out[26]= $-2\sqrt{5}$

Out[27]= $-2\sqrt{5}$

Out[28]= $2\sqrt{5}$

Out[29]= $2\sqrt{5}$

# CAS Exercise Examples for Chapter 12: Multiple Integrals

## ■ Section 12.1 Double Integrals

**Example:** Integrate $\int_0^1 \int_y^1 \frac{\sin x}{x}\, dx\, dy$ using *Mathematica*.

The easiest way to compute double integrals is to open the **BasicInput** palette and then click on the button containing $\int_{\square}^{\square} \square\, d\square$. Enter 0, then the Tab key and then enter 1. Press the Tab key again and then click on $\int_{\square}^{\square} \square\, d\square$ in the BasicInput palette. Enter y, then press the Tab key and then enter 1. Press the Tab key, enter the function $\frac{\text{Sin}[x]}{x}$, then press the tab key and enter x. Enter y after pressing the Tab key. You can then let *Mathematica* compute the integral.

```
In[18]:= ∫_0^1 ∫_y^1 Sin[x]/x dx dy

Out[18]= 1 - Cos[1]
```

If you want to compute a double integral by entering all your keystrokes from

the keyboard, enter the following.

```
In[19]:= Integrate[Integrate[Sin[x] / x, {x, y, 1}], {y, 0, 1}]

Out[19]= 1 - Cos[1]
```

## ■ Section 12.3 Double Integrals in Polar Form

**Example:** Given the double integral $\int_0^1 \int_{-y/5}^{y/5} x\sqrt{x^2 + y^2}\, dx\, dy$, complete each of the following.

(a) Plot the cartesian region of integration in the xy-plane.

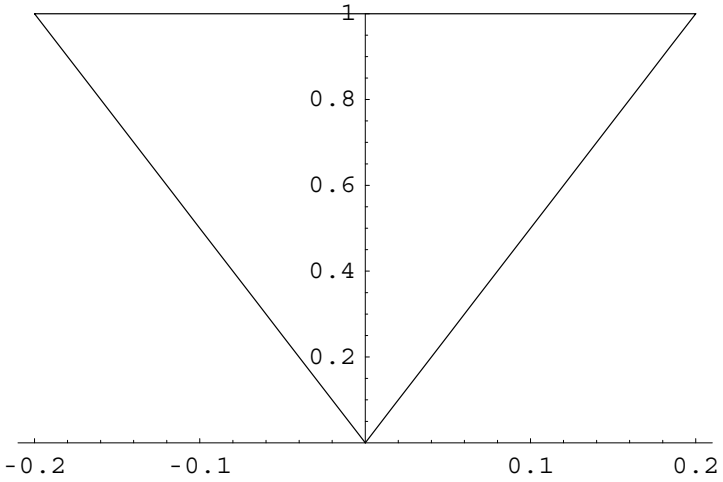
(b) Change each boundary curve of the Cartesian region in part (a) to its polar representation by solving its Cartesian equation for $r$ and $\theta$.

(c) Using part (b), plot the polar region of integration in the r$\theta$-plane.

(d) Change the integrand from Cartesian to polar coordinates. Determine the limits of integration from your plot in part (c) and evaluate the polar integral using *Mathematica*.

**Part (a)** Since $x = \pm y/5$ is equivalent to $y = \pm 5x$, the following `Plot` command can be used to plot the cartesian region of integration.

```
In[20]:= Plot[{5 x, 1, -5 x}, {x, -1/5, 1/5}, PlotRange -> {0, 1}];
```




**Part (b)** Replacing $y$ with $r\sin\theta$ and $x$ with $r\cos\theta$, the `Solve` command can then be used to find the values of r and $\theta$. (Ignore the warning messages.)

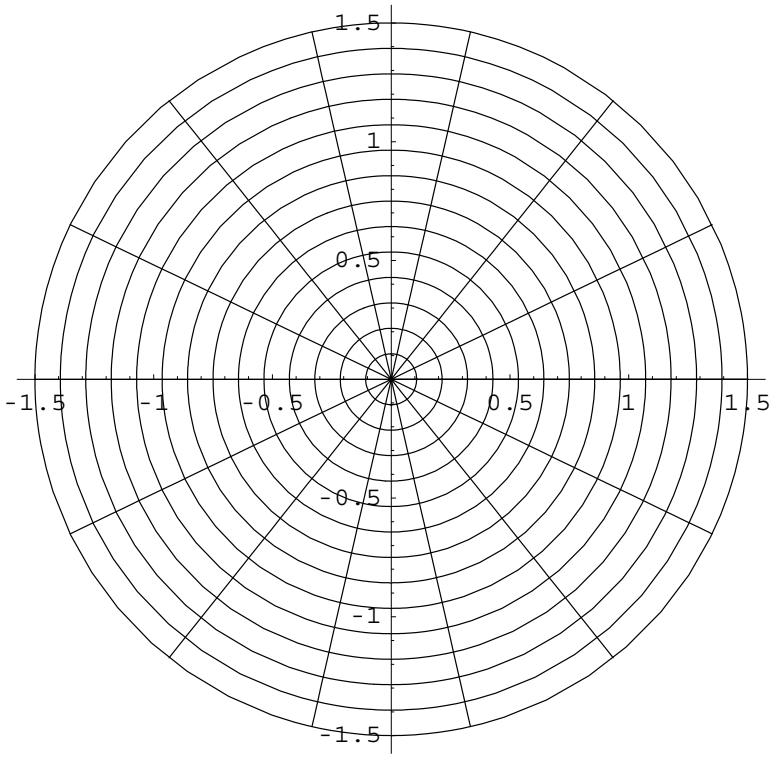
```
In[21]:= Solve[r Cos[θ] == -r Sin[θ] / 5, θ]
         Solve[r Cos[θ] == r Sin[θ] / 5, θ]
         Solve[r Sin[θ] == 1, r]

         Solve::ifun : Inverse functions are
            being used by Solve, so some solutions may  not be found.

Out[21]= {{θ → ArcCos[-1/Sqrt[26]]}, {θ → -ArcCos[1/Sqrt[26]]}}

         Solve::ifun : Inverse functions are
            being used by Solve, so some solutions may  not be found.

Out[22]= {{θ → -ArcCos[-1/Sqrt[26]]}, {θ → ArcCos[1/Sqrt[26]]}}

Out[23]= {{r → Csc[θ]}}
```

**Part (c):** The command `PolarPlot` is contained in the **Graphics** package and the package **ComplexMap** contains the command `PolarMap` used to plot a portion of the polar coordinate system.

```
In[24]:= << Graphics`Graphics`
         << Graphics`ComplexMap`
```

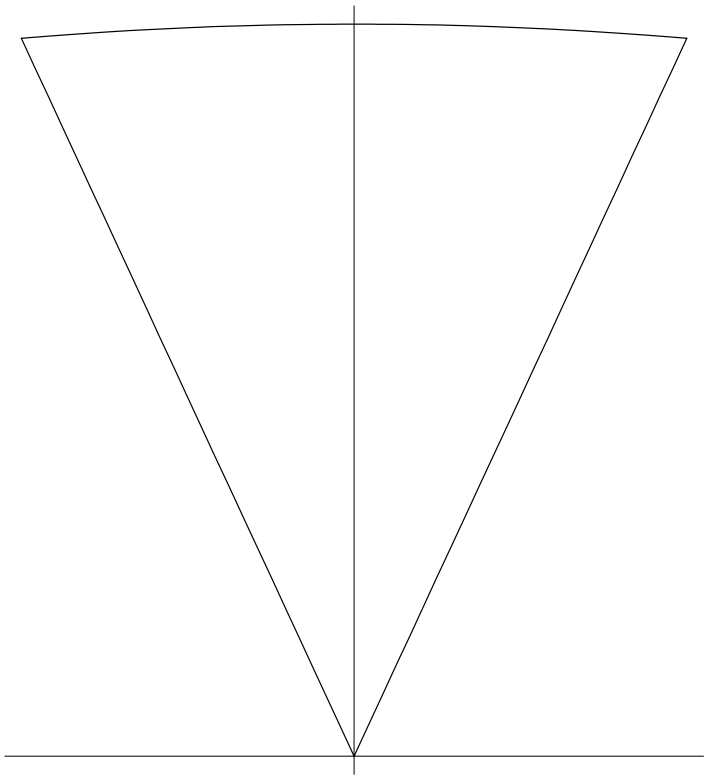
The following command will plot the polar coordinate system for $r = 0$ to 1.5.

In[26]:= **PolarMap[Identity, { 0, 1.5}];**



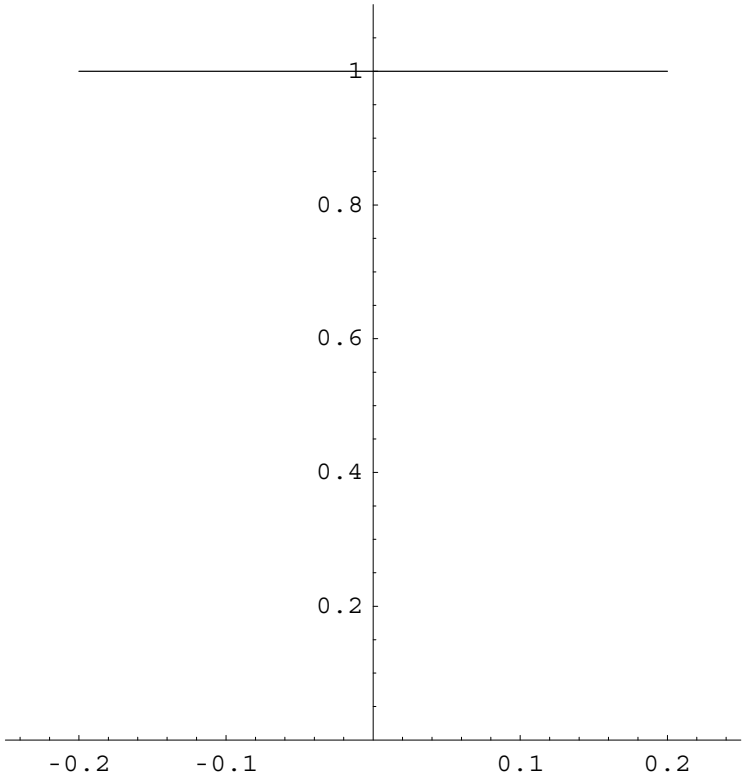

Since our region of integration is bounded by the lines $\theta$ =ArcCos$\left[\frac{1}{\sqrt{26}}\right]$ and $\theta$=ArcCos$\left[\frac{-1}{\sqrt{26}}\right]$, the next command is used to plot the portion of the polar coordinate system of interest.

In[27]:=
```
polargr = PolarMap[Identity, {0, 1.1, 1.1},
  {ArcCos[1/Sqrt[26]], ArcCos[-1/Sqrt[26]], ArcCos[-1/Sqrt[26]] - ArcCos[1/Sqrt[26]]},
  AspectRatio -> 1.1, Ticks -> None, GridLines -> None];
```



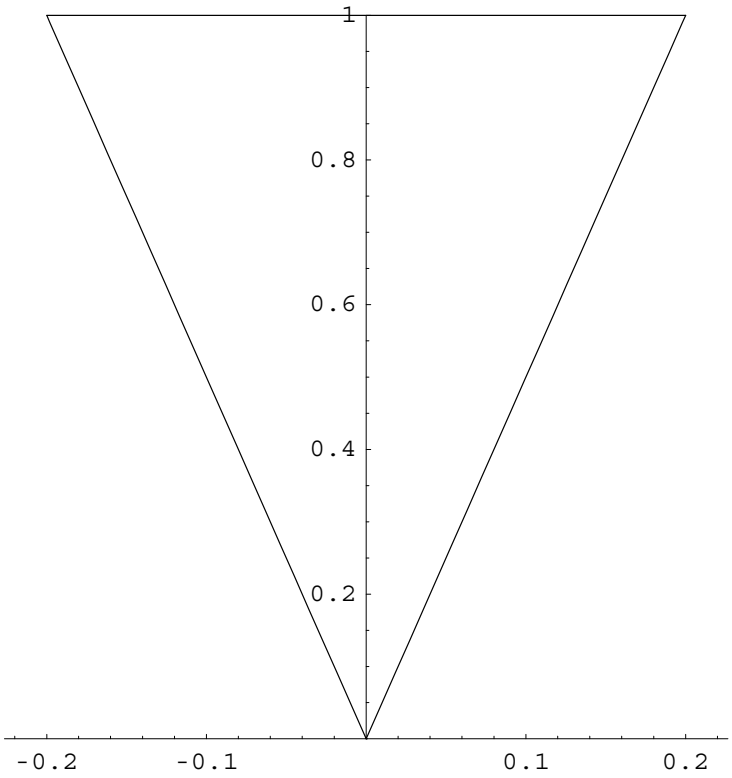
The upper bound on the region of integration is $r = \csc\theta$ and the corresponding graph of this function can be found using the `PolarPlot` command.

In[28]:=
```
upperbd = PolarPlot[Csc[θ], {θ, ArcCos[1/Sqrt[26]], ArcCos[-1/Sqrt[26]]},
    PlotRange → {{-.25, .25}, {0, 1.1}}, AspectRatio -> 1];
```




The region of integration can now be displayed in the polar coordinate system.

In[29]:=
```
Show[{upperbd, polargr}, PlotRange -> {0, 1}];
```




Part (d) The integral is now converted to polar form and evaluated.

In[30]:=
$$\int_{\text{ArcCos}\left[\frac{1}{\sqrt{26}}\right]}^{\text{ArcCos}\left[\frac{-1}{\sqrt{26}}\right]} \int_{0}^{\frac{1}{\sin[\theta]}} r^4 \cos[\theta]^2 \, dr \, d\theta$$

Out[30]=
$$-\frac{169\left(-\frac{7}{\sqrt{26}}+\cos\left[3\,\text{ArcCos}\left[-\frac{1}{\sqrt{26}}\right]\right]\right)}{25000}+\frac{169\left(\frac{7}{\sqrt{26}}+\cos\left[3\,\text{ArcCos}\left[\frac{1}{\sqrt{26}}\right]\right]\right)}{25000}+$$
$$\frac{1}{40}\log\left[\cos\left[\frac{1}{2}\,\text{ArcCos}\left[-\frac{1}{\sqrt{26}}\right]\right]\right]-\frac{1}{40}\log\left[\cos\left[\frac{1}{2}\,\text{ArcCos}\left[\frac{1}{\sqrt{26}}\right]\right]\right]-$$
$$\frac{1}{40}\log\left[\sin\left[\frac{1}{2}\,\text{ArcCos}\left[-\frac{1}{\sqrt{26}}\right]\right]\right]+\frac{1}{40}\log\left[\sin\left[\frac{1}{2}\,\text{ArcCos}\left[\frac{1}{\sqrt{26}}\right]\right]\right]$$

Now the numerical approximation of the integral is obtained.

In[31]:=
```
N[%]
```

Out[31]=
```
0.00107938
```

In order to check the answer, suppose you attempt to integrate in Cartesian

coordinates.

In[32]:=
$$\int_0^1 \int_{\frac{-y}{5}}^{\frac{y}{5}} x^2 \sqrt{x^2+y^2}\, dx\, dy$$

```
Integrate::gener : Unable to check convergence
```

Out[32]=
$$\frac{54\sqrt{26} + 625\,\text{Log}\left[-1+\sqrt{26}\right] - 625\,\text{Log}\left[1+\sqrt{26}\right]}{25000}$$

Despite the warning message, the solution obtained with *Mathematica* agrees with the earlier answer when the `N` command is used.

```
In[33]:= N[%]
Out[33]= 0.00107938
```

## ■ Section 12.4 Triple Integrals in Rectangular Coordinates

Evaluating triple integrals with *Mathematica* is completely analogous to computing double integrals. For example, the triple integral found in Example 3 in your textbook, is evaluated below.

In[34]:=
$$\int_0^1 \int_x^1 \int_0^{y-x} 1\, dz\, dy\, dx$$

Out[34]=
$$\frac{1}{6}$$

# CAS Exercise Examples for Chapter 13: Integration of Vector Fields

---

## ■ Section 13.1 Line Integrals

**Example:** Let $f(x, y, z) = \sqrt{1 + x^2 + y^2}$ and $\mathbf{r}(t) = t\mathbf{i} + t^2\mathbf{j} + t^3\mathbf{k}$, $0 \le t \le \frac{\pi}{2}$. Evaluate $\int_C f \, ds$ using Equation (2) in the text.

First, the function $f$ is defined along with the components of $\mathbf{r}(t)$.

```
In[1]:= f[x_, y_, z_] = Sqrt[1 + x^2 + y^2];
        {g[t_], h[t_], k[t_]} = {t, t^2, t^3};
```

Next, the integrand is formed and simplified.

```
In[3]:= integrand = f[g[t], h[t], k[t]] Sqrt[g'[t]^2 + h'[t]^2 + k'[t]^2] // Simplify
Out[3]= Sqrt[1 + t^2 + t^4] Sqrt[1 + 4 t^2 + 9 t^4]
```

If you try to evaluate the integral with *Mathematica*, an answer is not found. Therefore the `NIntegrate` command is used instead to obtain a numerical answer.

```
In[4]:= Integrate[integrand, {t, 0, Pi/2}]
Out[4]= Integrate[Sqrt[1 + t^2 + t^4] Sqrt[1 + 4 t^2 + 9 t^4], {t, 0, Pi/2}]

In[5]:= NIntegrate[integrand, {t, 0, 2}]
Out[5]= 27.5197
```

## ■ Section 13.2 Vector Fields, Work, Circulation, and Flux

**Example:** Find the work done by the force $\mathbf{F} = xz\,\mathbf{i} + z\mathbf{j} + yz\,\mathbf{k}$ over the curve $\mathbf{r}(t) = t^2\mathbf{i} + t\mathbf{j} + t^3\mathbf{k}$, $0 \le t \le 1$.

The package **VectorAnalysis** contains the command `DotProduct` which will compute the dot product of two lists of equal size. Study the following input and output used to solve the work problem.

```
In[6]:= << Calculus`VectorAnalysis`
```

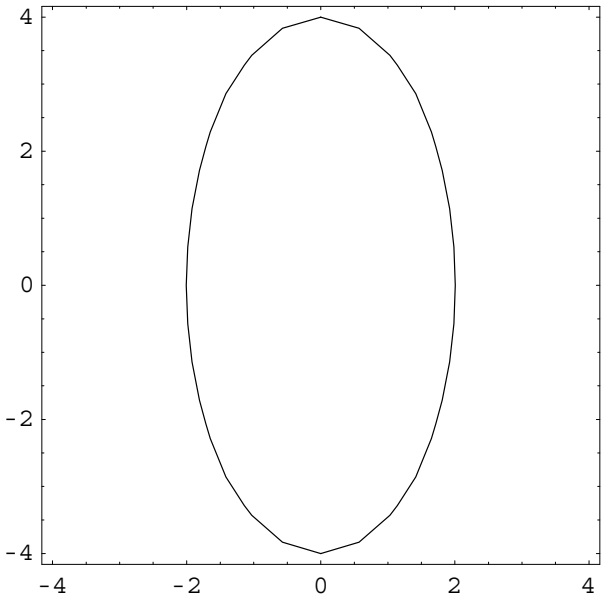
```
In[7]:= F[x_, y_, z_] = {x z, z, y z};
        {g[t_], h[t_], k[t_]} = {t^2, t, t^3};
        r[t_] = {g[t], h[t], k[t]};

In[10]:= F[g[t], h[t], k[t]]
Out[10]= {t^5, t^3, t^4}

In[11]:= r'[t]
Out[11]= {2 t, 1, 3 t^2}

In[12]:= integrand = DotProduct[F[g[t], h[t], k[t]], r'[t]]
Out[12]= t^3 + 5 t^6

In[13]:= ∫_0^1 integrand dt
Out[13]= 27/28
```

## ■ Section 13.4 Green's Theorem in the Plane

**Example:** Find the counterclockwise circulation of the field $\mathbf{F} = (x - 2y)\mathbf{i} + (3x + y)\mathbf{j}$ around the simple closed curve C: $4x^2 + y^2 = 16$.

`ContourPlot` is used to see the C.

```
In[14]:= ContourPlot[4 x^2 + y^2, {x, -4, 4},
           {y, -4, 4}, Contours -> {16}, ContourShading -> False];
```



The second form of Green's Theorem in equation (4) in your text is used here to compute the circulation.

In[15]:=
```
m[x_, y_] = x - 2 y; n[x_, y_] = 3 x + y;
```

In[16]:=
```
integrand = ∂x n[x, y] - ∂y m[x, y]
```

Out[16]= 5

In[17]:=
$$\int_{-2}^{2}\int_{-\sqrt{16-4x^2}}^{\sqrt{16-4x^2}} \text{integrand}\, dy\, dx$$

Out[17]= $40\pi$