Maple in Action

An Introductory Handbook

Piroz Mohseni

Iowa State University

Forward

My first introduction to Computer Algebra Systems came during a differential equations course at Iowa State University. From the start, the power of this class of software impressed me. The instructor intentionally made the numbers "unpleasant" to encourage usage of a CAS to do some of the assignments.

This handbook is intended for all engineering and science students or instructors, who are interested in learning *Maple*. We start by a broad introduction to the syntax of *Maple* along with a number of simple examples to familiarize the user with the system.

The rest of the handbook is what I hope the reader will find unique. A total of fifteen problems from engineering and science textbooks are presented. These are the types of problems that an undergraduate student will face throughout his/her college classes. In each problem, *Maple* is used to find a solution.

My philosophy is that a user can find "dry" examples in a reference manual or the on-line help pages. The steepest learning curve for a system like *Maple* is how to put everything together and solve a complete problem. In many of the problems, the reader is encouraged to use the trial and error method. Such approach to learning is less costly with the help of a CAS, since lengthy mathematical calculations are done by the computer and the user can concentrate on the physical problem itself.

I have used *Maple* extensively throughout my undergraduate classes and I found it extremely helpful. I encourage all science and engineering students to give *Maple* a try. It can be a great friend during those long semester projects or even during your daily assignments. I hope that you find this handbook useful in your journey to the world of computer algebra systems. I welcome any comments or suggestions you have with regards to this project. Good luck to you and enjoy.

Piroz Mohseni
January 1995

Contents

List of Figures

1. Getting Started with *Maple*

Overview

Maple is a comprehensive general purpose computer algebra system. It is used primarily in education, scientific research, mathematics and engineering. *Maple* can do both numerical and symbolic calculations and has facilities for 2 and 3-dimensional graphical output. *Maple* is also a programming language which supports procedural and functional programming. This introduction is written using *Maple* V, Release 2. On a UNIX machine, you can start *Maple* by typing:

`maple` (for the text version), or
`xmaple` (for the X Window version)

You should get something similar to the following message for the text version.

```
    |\^/|      Maple V Release 2 (Iowa State University)  .
._|\|   |/|_.  Copyright (c) 1981-1992 by the University of Waterloo.
 \  MAPLE  /   All rights reserved. MAPLE is a registered trademark of
 <---------->  Waterloo Maple Software.
      |        Type ? for help.
```

Basic Syntax

The *Maple* prompter is >. In this mode, you may enter commands, define variables and perform calculations. One of the most common mistakes new users make is that they forget to end the command with a valid termination character. If a command is terminated with a semicolon, then the result of that command is shown on the screen. If a command is terminated with a colon, then the result is not shown. *Maple* is case sensitive so `diff()` and `Diff()` are interpreted as different functions. Comments start with `#` and continue to the end of line.

On-line Help

Maple contains an extensive on-line help facility which contains descriptions and examples for almost all the supported functions and packages. To access the help information about a particular item, type a question mark followed by the name of the item (e.g., to get help about the integration function type `?int`) Here is the help screen for the square root function sqrt as an example. Enter `?sqrt` to access this screen.

```
FUNCTION: sqrt - square root

CALLING SEQUENCE:
   sqrt(x)

PARAMETERS:
   x - any algebraic expression

SYNOPSIS: - The function sqrt forms the square root of its argument x.
It returns the square root of integers and polynomials which are
perfect squares.  It is mapped onto products.

EXAMPLES:
> sqrt(3);
                                          1/2
                                         3

> sqrt(4);
                                          2

> sqrt(12);
                                           1/2
                                        2 3

> sqrt(3+4*I);
                                        2 + I
> sqrt(x^2/9);
                                        1/3 x

> sqrt(a);
                                          1/2
                                         a

SEE ALSO:  simplify[sqrt], inifcns
```

Description of various parts of the help screen follow.

- FUNCTION

  The mathematical name of the function is mentioned here so the user immediately knows whether the given function is what he or she was looking for.

- CALLING SEQUENCE

  The general form of the function along with placement of the parameters is shown here. Some functions can be called in several different form, so a function could have more than one entry under this section.

- PARAMETERS

  Each of the parameters, if any, of the function are described here. This is important since an error will occur if the wrong parameter type is passed to the function. For example, if the sine function expects a parameter in radians, then providing a parameter is degrees will lead to erroneous results.

- SYNOPSIS

  This section includes a description of the function and any miscellaneous notes which the user might find useful. Many of your questions are answered in this section.

- EXAMPLES

  A number of examples along with their *Maple* output are shown here. The user can see the different ways the function can be used along with the result.

- SEE ALSO

  This part lists the names of some other functions which somehow relate to the current function.

Mathematical Notation

The basic arithmetic operators and constants known to *Maple* are shown in Table 1.1.

Using *Maple* Interactively

- Numerical Capabilities

  Maple is capable of performing numerical calculations just like a calculator but with greater precision. To perform simple arithmetic, just type in the numbers and end the line with a semicolon. The result will be displayed.

  ```
  > 6785*3432;
  >
                 23286120
  ```

Maple notation	Meaning	Mathematical notation
x+y	addition	$x + y$
x-y	subtraction	$x - y$
x*y	multiplication	$x * y$
x/y	division	x/y
x ^ y or x**y	exponentiation	x^y
k!	factorial	$k!$
I or sqrt(-1)	complex unit	i or $\sqrt{-1}$
Pi		π
infinity		∞
abs(x)	absolute value	$\|x\|$
sqrt(x)	square root	$\sqrt{x}$
exp(x)	exponential function	e^x
ln(x)	natural logarithm	$\ln x$
sin(x)	sine function	$\sin x$
cos(x)	cosine function	$\cos x$
tan(x)	tangent function	$\tan x$

Table 1.1: Common Mathematical notations.

Due to the symbolic nature of *Maple*, integer arithmetic will be done exactly. For example:

```
> 23/34;
                         23
                        ----
                         34
```

To get a real result, use the `evalf()` function, or include a decimal point with the number.

```
> evalf(23/34);

                  .9583333334

> 23./24;

                                          .9583333334
```

Here is another example:

```
> sqrt(65);
>
                          1/2
                        65

> evalf(");
>
                  8.062257748

> sqrt(65.0);

                                          8.062257748
```

Note that " refers to the result of the *previous* calculation.

Maple can display numbers with unlimited precision. Of course, higher precision means slower speed and consumption of more memory resources. Here is the number π up to 15 decimal points:

```
> evalf(Pi,15);
>
                   3.14159265358979
```

- Mathematical Functions

Just like a calculator, *Maple* supports all of the common mathematical functions. In some cases the name of the function might be slightly different, but usually the name of the mathematical function is the same. Here are some common functions and how to use them. Of course, detailed help is available for all the mathematical functions.

```
> sin(Pi/2);
>
                  1

> ln(12.3);
>
                2.509599262
```

- Symbolic Capabilities

  Maple is a Computer Algebra System and thus contains a number of symbolic functions. The ability to manipulate mathematical entities in a symbolic way is probably the most distinguishable feature of *Maple*. Here are some elementary examples to get you started. More examples can be found in the help screens of the particular function.

  - **Integration**

    ```
    > int(3*x^2,x);
    >
                     3
                    x
    ```

  - **Differentiation**

    ```
    > diff(sin(5*x),x);
    >
              5 cos(5 x)
    ```

    There are several ways to obtain higher order derivatives. Here is an example for a second order derivative:

    ```
    > diff(x^3,x);

                                          2
                                       3 x

    > diff(x^3,x$2);

                                       6 x

     diff( diff(x^3,x),x );

                                       6 x
    ```

  - **Factoring**

    ```
    > factor(x^2-2*x-15);
    >
          (x + 3) (x - 5)
    ```

– **Linear equations**

```
> solve({x+y=1, 2*x+y=3}, {x,y});
>
       {y = -1, x = 2}
```

– **Differential equations**

We define the differential equation $y'' + 5y' + 6y = 0$.

```
>eq1:= diff(y(x),x$2) + 5*diff(y(x), x) + 6*y(x) = 0;

                     /   2       \
                     |  d        |     /  d      \
          eq1 :=     |----- y(x)| + 5 |---- y(x)| + 6 y(x) = 0
                     |   2       |     \ dx      /
                     \ dx        /
```

Use the function dsolve to solve the differential equation:

```
> dsolve(eq1, y(x));

             y(x) = _C1 exp(- 2 x) + _C2 exp(- 3 x)
```

Unless initial conditions are specified, *Maple* will generate a solution with constants `_C1, _C2`, etc. To specify initial conditions, enclose the definition of the equation and its initial conditions in braces as shown below:

```
>dsolve({eq1, y(0)=0, y(1)=2},y(x));

                       exp(- 2 x)                exp(- 3 x)
         y(x) = 2 ----------------- - 2 -----------------
                   exp(-2) - exp(-3)      exp(-2) - exp(-3)
```

Graphics Capabilities

Maple contains a wide variety of plotting capabilities from simple 2-dimensional plots to sophisticated 3-dimensional animation. Ample examples are given in the tutorial and the help screens. The simplest plotting command is `plot(f(x), x=a..b)` where f(x) is the function to be plotted in terms of the variable x. The constants a and b specify the range of the plot. Several graphical examples are shown throughout the problems in this handbook.

Packages

Every time you run *Maple*, a considerable amount of memory is devoted to the built-in functions and the routines to perform arithmetic and symbolic evaluation. Due to the limitation of memory, many of *Maple*'s functions are grouped into units called packages and stored on disk. Whenever the user needs these functions, they can be called using the `with(package_name)` command. A common example is the linear algebra package.

Internet #	Internet Address	Location	Directory
129.97.140.58	daisy.uwaterloo.ca	Canada	pub/maple
192.93.2.54	ftp.inria.fr	France	lang/maple
192.16.187.3	ftp.can.nl	Holland	pub/maple-ftplib
129.132.101.33	neptune.inf.ethz.ch	Zurich	pub/maple
192.139.233.5	ftp.maplesoft.on.ca	Canada	pub/maple/share

Table 1.2: FTP sites for Maple related items.

This package contains functions to manipulate matrices and vectors. To use these functions one issues the following command:

```
with(linalg);
```

A list of the functions which are included in the package is displayed. You can then call the function like any of the *Maple* functions.

Where to Get More Information

- Books & Journals

  One of the best introductory books for *Maple* is "First Leaves: A Tutorial Introduction to *Maple* V" from Springer-Verlag. Two other classic books are "*Maple* V Library Reference Manual" and "*Maple* V Language Reference Manual." Also, *MapleTech* is a journal devoted to *Maple*. You can contact Springer-Verlag at (800)-SPRINGER for more information.

- USENET News

  If you subscribe to USENET News, the news group `sci.math.symbolic` is devoted to discussions about Computer Algebra systems including *Maple*.

- FTP sites

  Waterloo Software maintains an ftp site which holds contributions from users. There are some introductory text written by users which you could obtain. The ftp addresses are shown in table 1.2.

- Waterloo Software *Maple* originated at the University of Waterloo by the Symbolic Algebra Group. Product information can be obtained from Waterloo Maple Software by calling 1-800-267-6583 or send email to `info@maplesoft.on.ca`. There is also a WWW Homepage for *Maple*. Its URL is `http://daisy.waterloo.edu`.

2. Solving linear equations

In this problem we will use *Maple* to solve a resistive circuit problem. Consider the circuit shown in Fig. 2.1. Three mesh current equations are used to describe the circuit. These are three linear equations and the methods to solve such systems are well covered in introductory Algebra text. *Maple* provides an easy way to find the solution and it works no matter how tedious the number crunching is. It turns out that a single function called `solve` is all that is needed to find a solution.

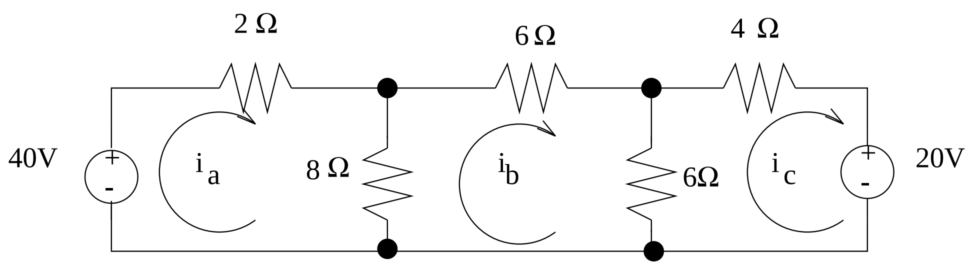

Figure 2.1: Simple resistive circuit.

We first define the equations.

```
> eq1:= -40+2*ia + 8*(ia - ib) = 0;
                           eq1 := - 40 + 10 ia - 8 ib = 0

> eq2 := 8* (ib - ia) + 6*ib + 6 * (ib-ic) = 0;
                           eq2 := 20 ib - 8 ia - 6 ic = 0

> eq3 := 6 * (ic-ib) + 4*ic + 20 = 0;
                            eq3 := 10 ic - 6 ib + 20 = 0
```

We can now use the `solve` function. We give it two sets: a set of the equations and a set of variables to solve for. These two sets should have the same number of elements.

```
> solve({eq1, eq2, eq3}, {ia, ib, ic});
                           {ic = -4/5, ia = 28/5, ib = 2}
```

There are a number of variations to the `solve` command for solving different kinds of systems and applying different methods to find the solution set. Examples include `dsolve, fsolve, isolve`, and `msolve`. This handbook covers a number of these variations.

3. Differentiation

An essential tool for analyzing a number of engineering problems is the ability to analyze functions and their derivatives. To give some physical sense to the problem, we use a potential energy function in this problem.

The potential energy function for the force between two atoms in a diatomic molecule can be expressed approximately as follows:

$$U(x) = \frac{a}{x^{12}} - \frac{b}{x^6}$$

where a and b are positive constants and x is the distance between atoms. We can represent this function in *Maple* using the function notation as follows:

```
U:= x-> a/x^12 - b/x^6;
```

Part 1

At what values of x is $U(x)$ equal to zero? At what values of x is $U(x)$ a minimum?

The first part of the question can be answered by setting $U(x)$ equal to zero and solving for x in terms of a and b. The general purpose function `solve` can help us do this.

```
> solve(U(x)=0, x);
         1/6                  1/2   1/6                          1/2   1/6     1/6
        a     (1/2 + 1/2 I 3   ) a     (- 1/2 + 1/2 I 3   ) a       a
        ----, -----------------------, -------------------------, - ----,
         1/6              1/6                          1/6              1/6
        b               b                          b               b

                              1/2   1/6                        1/2   1/6
           (- 1/2 - 1/2 I 3   ) a     (1/2 - 1/2 I 3   ) a
           -------------------------, -----------------------
                         1/6                        1/6
                        b                          b
```

Note that *Maple* has computed complex solutions as well as real solutions. The setting of the problem mandates a real, positive solution. Therefore our solution is $(\frac{a}{b})^{\frac{1}{6}}$.

To find the minimum we ask *Maple* to solve for when the first derivative of the function equals to zero.

```
> solve(diff(U(x), x)=0, x);
   1/6  1/6                 1/2   1/6  1/6                        1/2   1/6  1/6
  2    a     (1/2 + 1/2 I 3   ) 2    a     (- 1/2 + 1/2 I 3   ) 2    a
  ---------, ----------------------------, ------------------------------,
      1/6                  1/6                                1/6
     b                    b                                  b
```

```
   1/6  1/6                               1/2    1/6  1/6                              1/2    1/6  1/6
  2    a       (- 1/2 - 1/2 I 3    ) 2     a      (1/2 - 1/2 I 3    ) 2     a
- ---------, -----------------------------, ----------------------------
     1/6                     1/6                          1/6
    b                       b                            b
```

Once again, it is up to us to select the answer which applies to the physical setting of our problem. In this case it would be $\frac{\sqrt[6]{2}\sqrt[6]{a}}{\sqrt[6]{b}}$.

Part 2

Determine the force between the atoms.

We know the following relationship (for an explanation refer to your favorite physics text).

$$F(x) = -\frac{d}{dx}U(x)$$

We use the `diff` function to find the first derivative of U(x).

```
> -diff(U(x), x);
                          a        b
                    12 ---- - 6 ---
                         13       7
                        x        x
```

Part 3

The energy needed to break up the molecule into separate atoms is called the dissociation energy. What is the dissociation energy of the molecule?

If one atom has enough kinetic energy to get over the potential hill, it will not longer be bound to the other atom. Hence, the dissociation energy D equals the change in potential energy from the minimum value to the value at x=∞.

```
> simplify(U(infinity) - U( (2*a/b)^(1/6)));
                            2
                           b
                     1/4 ----
                            a
```

4. Solution to ordinary differential equations

Differential equations describe many physical models including electrical circuits. An understanding of differential equations and their symbolic or numeric solutions is very important in many engineering disciplines. *Maple* has an extensive engine for dealing with differential equations.

In the following example, we analyze the natural response of an RC circuit. We describe the behavior of the circuit in terms of a differential equation and then we find a symbolic solution for it.

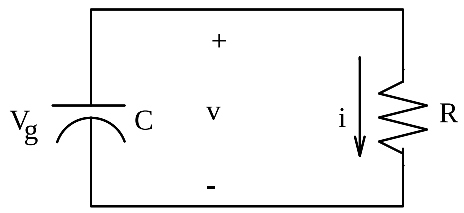

Figure 4.1: A RC circuit after switching.

Consider the circuit shown in Fig. 4.1. Using the lower junction between R and C as the reference node and summing the currents away from the upper junction between R and C gives us a differential equation for the circuit. We define it in *Maple* like the following:

```
> eq1:= diff(v(t), t) * C + v(t)/R = 0;
                                 /  d       \      v(t)
                         eq1 := |---- v(t)| C + ---- = 0
                                 \ dt       /       R
```

We denote the initial voltage of the capacitor as `Vinit` and thus our initial conditions is:

```
> eq2:= v(0) = Vinit;
                              eq2 := v(0) = Vinit
```

We now have all the pieces *Maple* needs to solve this differential equation. We use the `dsolve` function to find a solution.

```
> dsolve({eq1, eq2}, v(t));
                                            t
                           v(t) = exp(- ---) Vinit
                                           C R
```

As you can see, *Maple* has found a solution. Note that so far we have not used any numbers. Everything has happened symbolicly which gives us a nice interface to find general and analytical solutions.

Using Ohm's law, we want to find an expression for the current `i(t)`. This requires a simple calculation of dividing the answer we got for `v(t)` by the resistor `R`.

```
i:="/R;
                                    t
                           exp(- ---) Vinit
                   v(t)          C R
               i := ---- = ----------------
                    R              R
```

Recall that " refers to the result from the previous calculation. As you can see, *Maple* has simply divided both sides of the equation by the value `R`. The answer which is assigned to the variable `i` is an *equation* not an *expression*. This will create problems later when we try to differentiate or integrate `i`. We somehow need to assign to `i` the expression (right hand side of the equation.) To do that we use the `op` function.

```
> v_t:= op(2,");

                                  t
                 v_t := exp(- ---) Vinit
                                 C R
```

Here we are instructing *Maple* to grab the second operand from the last expression and assign it to `v_t`. The `op` function is a very useful tool for symbolic manipulation. Learn how it works if you plan to do extensive symbolic calculations with *Maple*.

Now that we have an expression for the voltage, we can find the current easily.

```
> i:= v_t /R;
                                  t
                         exp(- ---) Vinit
                                 C R
                   i := ----------------
                                   R
```

By multiplying voltage and current, we can find the power. Note the use of the `simplify` command. Try the following expression both with and without the `simplify` function and see the difference.

```
> p:=simplify( v_t * i);
                                  t        2
                       exp(- 2 ---) Vinit
                                 C R
                 p := -------------------
                                  R
```

Finally, we find an expression for the energy dissipated in the resistor. Mathematically we want to do the following:

$$w = \int_0^t p dx = \int_0^t e^{-\frac{2x\,t}{CR}}\, Vinit^2 R^{-3} dx$$

Note the introduction of the dummy integration variable `x`. Although such treatment is possible within *Maple*, an easier way is to change the name of the upper bound integral variable to something other than `t`. We have chosen the name `tr`.

```
> w := int(p, t=0..tr);
                                   tr       2                2
             w := - 1/2 C exp(- 2 ---) Vinit  + 1/2 C Vinit
                                  C R


> factor(w);
                                  2 /         tr     \
                    - 1/2 C Vinit  |exp(- 2 ---) - 1|
                                    \        C R     /
```

5. Waves

Waves and their properties can cause confusion since they contain a time dimension along with the ordinary spatial dimensions. Many texts attempt to address this problem by displaying a snapshot of the wave at different time intervals. With *Maple* you can view a continuous animation of the wave and observe the different phenomena such as standing waves. Aside from a few basic plots, we will not display snapshots of the wave.The reader is encouraged to try the examples in this section on a computer and view the animations.

We define two waveforms and their sum as follows:

```
> y1:= ym * sin (k * x - w*t - phi);
                          y1 := ym sin(k x - w t - phi)

> y2:= ym * sin(k*x - w*t);
                             y2 := ym sin(k x - w t)

> y := simplify(y1+y2);
                 y := ym sin(k x - w t - phi) + ym sin(k x - w t)
```

Here `k` is the wavenumber, `w` is the angular frequency, `ym` is the amplitude, `t` is time, `x` is position (one-dimension), and finally `phi` is the phase constant.

Since we are interested in graphical results, we must assign some numerical values to the constants in our wave equations.

```
> ym:= 1;
                                    ym := 1

> k:= 1;
                                    k := 1

> w:= 2*Pi;
                                   w := 2 Pi

> t:= 0;
                                    t := 0

> phi:= Pi/12;
                                phi := 1/12 Pi
```

To see how the phase constant changes the waveform and to look at the sum of the waves, we plot the three waveforms for t=0. The result is shown in Fig. 5.1.

```
> plot({y1, y2, y}, x=0..10);
```

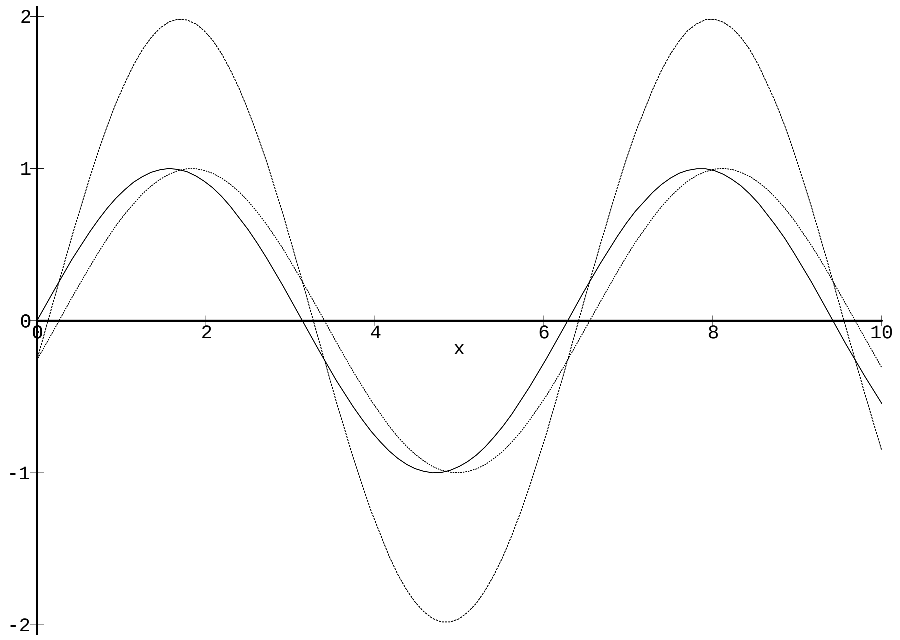

Figure 5.1: Plot of waveforms `y1, y2` and their sum.

Now let us increase the phase difference and see the result. What would you predict will happen to the plot in Fig. 5.1?

```
> phi := 11*Pi/12;

                                   11
                           phi := ---- Pi
                                   12

> plot({y1, y2, y}, x=0..10);
```

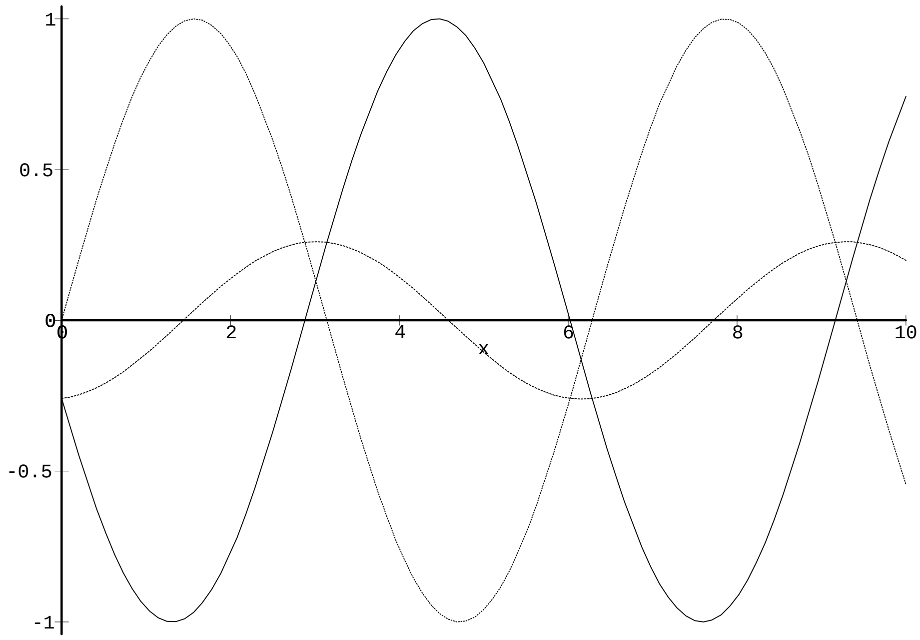

Figure 5.2: Plot of waveforms with an increased phase constant.

We are now ready for animation. The `animate` function is defined within the `plots` package. Before performing the animations, we must restore the variable `t` to its original value. Remember that earlier we set the value of `t` to zero. Since time will be a variable in animations, we must restore `t` to be a variable with no assigned value.

```
> with(plots):
> t:= 't';
                                        t := t
> animate({y1,y2,y}, x=0..10, t=0..5);
```

Next we animate standing waves. In a standing wave, a particle at any particular point x executes simple harmonic motion as time goes on, and all particles vibrate with the same frequency.

```
>
# Standing Waves
> y1:= ym * sin (k * x - w*t);
                                y1 := sin(x - 2 Pi t)
> y2:= ym * sin (k * x + w*t);
                                y2 := sin(x + 2 Pi t)
> y:= y1+y2;
                       y := sin(x - 2 Pi t) + sin(x + 2 Pi t)

> animate({y1,y2,y}, x=0..10, t=0..10, frames=60);
```

The reader is encouraged to experiment with different waveforms and observe the effects as wave parameters vary. There is also a facility to animate three-dimensional waves using the `animate3d` function.

6. Transistor characteristic curves

Maple offers an extensive set of plotting functions. Graphs are a powerful tool to view the results of an investigation or calculation. Graphs provide an "overall" picture of the results and contain an extensive amount of information. In the following problem we will graph the characteristic function of a MOS transistor. The reader should be reminded that unfamiliarity with the setting of the problem should not cause any worries. The methods are important.

We define the characteristic function externally. That is we create a plain text file containing the definition of our function. For a simple case such as ours, this method might seem time consuming, but for more sophisticated functions, this is an easy method to reuse popular function definitions.

Use your favorite editor and create a text file containing the following:

```
Ids:=proc(Vgs, Vt, Vds, B)
if (Vgs < Vt) and (Vds >= 0) then
  RETURN(0);
elif (Vgs > Vt) and (Vds > 0) and (Vds <= (Vgs - Vt)) then
RETURN (B*(Vgs - Vt) * Vds - (Vds^2/2));
elif (Vgs > Vt) and (Vds > (Vgs - Vt)) then
RETURN (B/2 * (Vgs - Vt)^2);
fi;
end;
```

Here we are defining a function called `Ids`. In *Maple's* terminology, it is called a *procedure*. We pass to this procedure four parameters: `Vgs, Vt, Vds` and `B`. The body of the function checks these parameters to see in which region the transistor is operating (namely saturated, ohmic, or off) and then performs the appropriate calculation and returns `Ids`. [1] If you know any programming languages, the syntax should be familiar to you.

Suppose you have saved the above function in a file named `trans.model`. To read it into *Maple* issue the following command. Note the usage of back quotes inside the `read` function.

```
> read(`trans.model`);
Ids := proc(Vgs,Vt,Vds,B)

            if Vgs < Vt and 0 <= Vds then RETURN(0)
            elif Vt < Vgs and 0 < Vds and Vds <= Vgs-Vt then
                RETURN(B*(Vgs-Vt)*Vds-1/2*Vds^2)
            elif Vt < Vgs and Vgs-Vt < Vds then RETURN(1/2*B*(Vgs-Vt)^2)
            fi
        end
```

[1] This function ignores the Lambda effects.

Now the function `Ids` is defined within the *Maple* environment. Try a test call to this procedure like the following:

```
> Ids(5, 1, 3, 1);
                                        15/2
```

We are now ready to graph the function using the `plot` command.

```
>plot({'Ids(5, 1, Vds,1)','Ids(4,1,Vds,1)','Ids(3, 1, Vds,1)','Ids(2,
1, Vds,1)','Ids(1, 1, Vds,1)','Ids(0, 1, Vds,1)'} , Vds=-1..7);
```

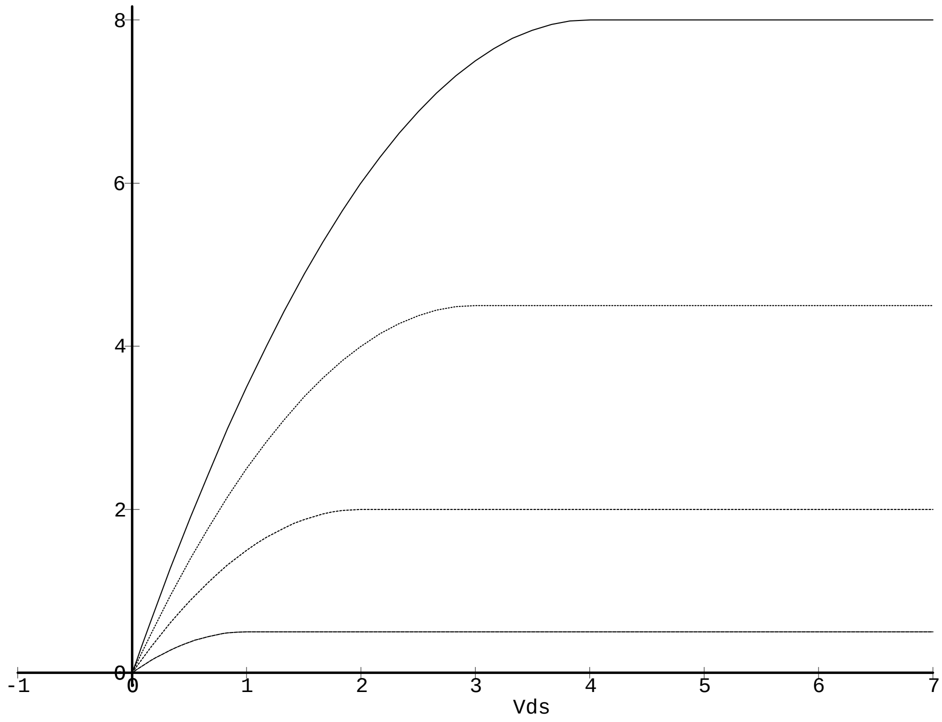

Figure 6.1: Characteristic Curve for a transistor

Here we are plotting the same function for different values of `Vgs` to get the characteristic curve of the transistor. Note that we are enclosing each call to the function in single quotes. This need arises from the way *Maple* treats procedure calls within another procedure. In our case we are calling the `Ids` function within the `plot` function.

One interesting aspect of a program like *Maple* is that it allows the user to vary parameters and experiment. The standard treatment of transistors is that of using characteristics curve like what is shown above. In this curve, we are actually plotting three parameters in two dimensions. We keep `Vgs` constant and plot `Ids` vs. `Vds`. With little work, we can plot the three parameters in three dimensions using the `plot3d` command.

```
>plot3d('Ids(Vgs, 1, Vds, 1)', Vgs=0..5, Vds=-1..7);
```

This command will create a three dimensional plot in which all three parameters `Vgs, Vds` and `Ids` vary. For some students this plot gives a better picture of how a transistor operates. Note that we can use the mouse to easily change the orientation of the graph. A few examples follow.

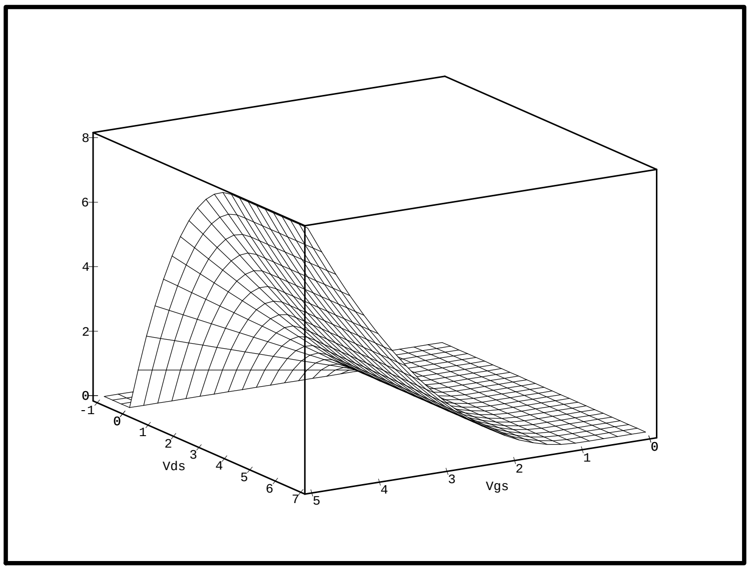

Figure 6.2: 3-dimensional characteristic curve.

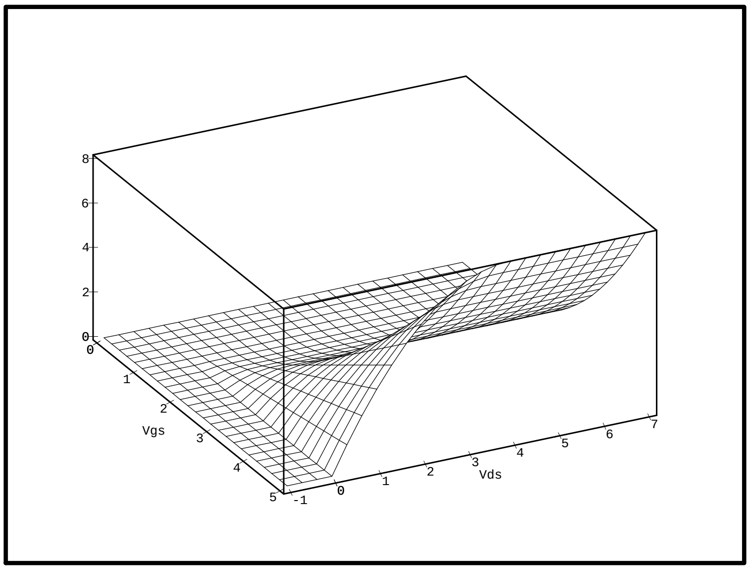

Figure 6.3: 3-dimensional characteristic curve.

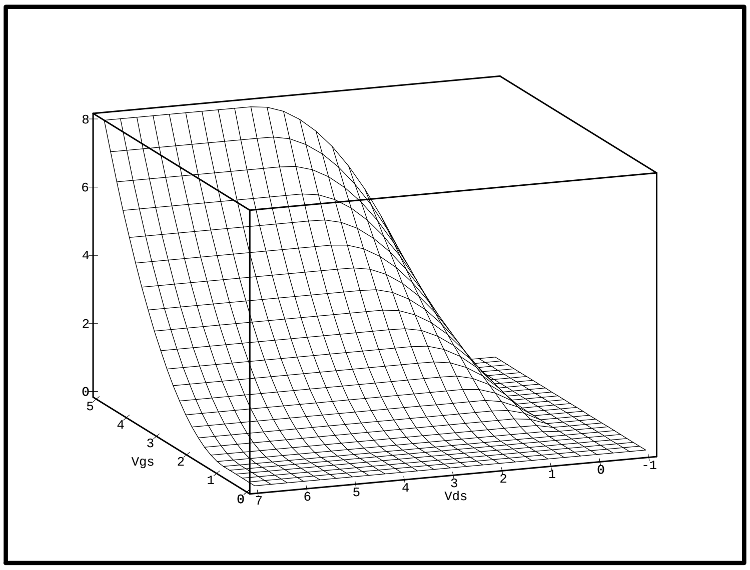

Figure 6.4: 3-dimensional characteristic curve.

7. Complex Numbers

There are a number of applications in engineering that require manipulation of imaginary numbers. Despite the simplicity of the algebraic rules for complex numbers, one can easily make arithmetic errors when dealing with relatively large systems. In this exercise, we look at how *Maple* handles complex numbers. The imaginary number i or j is denoted by a capital `I` in *Maple*. For example, to denote the number $j52$, we use the notation `52 * I`.

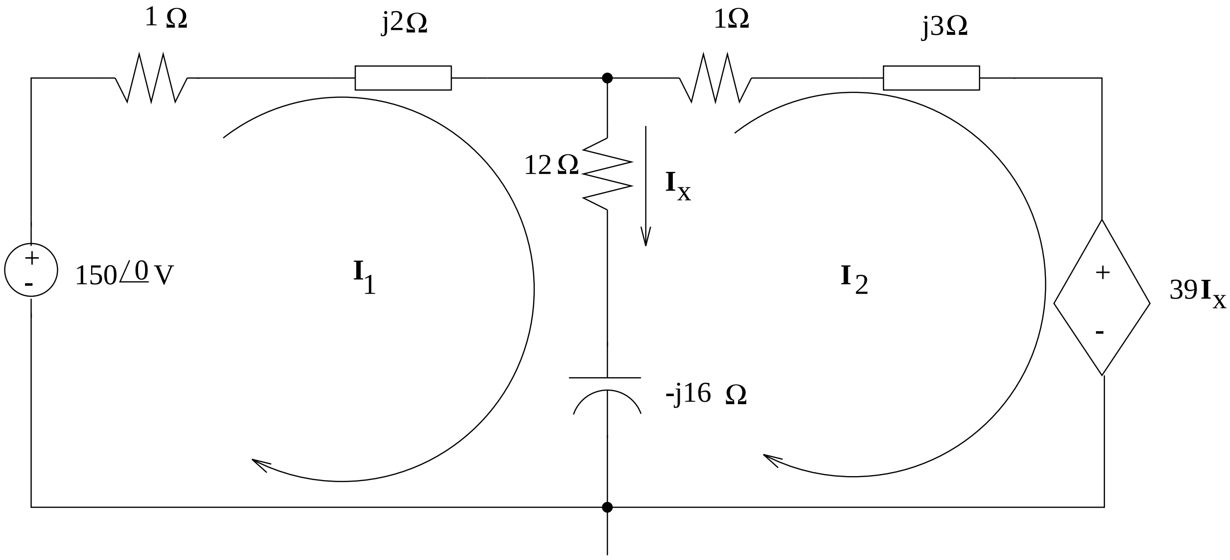

Figure 7.1: Circuit for problem 7.

Consider the circuit in Fig. 7.1 Let us use the mesh current method to find currents I_1, I_2 and I_x. Here are the three equations:

```
> eq1:=(1+2*I)*i1 + (12-16*I)*(i1-i2) = 150;

              eq1 := (1 + 2 I) i1 + (12 - 16 I) (i1 - i2) = 150

> eq2:= (12-16*I)*(i2-i1) + (1+3*I)*i2 + 39*ix = 0;

            eq2 := (12 - 16 I) (i2 - i1) + (1 + 3 I) i2 + 39 ix = 0

> eq3:= ix = i1-i2;

                          eq3 := ix = i1 - i2
```

As usual, we can use the `solve` function to find a solution. Note that we are assigning the solution set to the variable `s`.

```
> s:=solve({eq1,eq2,eq3}, {i1,i2,ix});

            s := {ix = - 2 + 6 I, i2 = - 24 - 58 I, i1 = - 26 - 52 I}
```

We now have a solution set `s` with three variables `ix`, `i2` and `i1`. However, the variables really don't have any values assigned to them. We can see that by simply typing the variable name followed by a semicolon.

```
> i1;
                          i1;
```

This behavior would not pose any problems unless we plan to use the values of the variables in future calculations. The `assign` command can be used to perform the three assignments for us (one for each variable).

```
> assign(s);
```

Prior to ending this exercise, let us explore one other useful function for complex arithmetic. In some cases, we need to convert a complex expression from its coordinate form to its polar form and vice versa. To do this, we first need to load a library function as follows:

```
> readlib(polar);
proc(r,th) ... end
```

We can now use the `convert` function to change the complex variables from their coordinate form to polar form.

```
> convert(v1, polar);
                         polar(70.43294684, -.2408464552)

> convert(v2,polar);
                         polar(72.80109892, -.3652014498)

> convert(ix,polar);
                         polar(4.118252057, .4201967127)
```

One last thing about this type of conversion. Consider the following conversion:

```
> convert(3+4*I, polar);
                              polar(5, arctan(4/3))
```

Suppose we want a number rather than the expression `arctan(4/3)`. This can be done via the `map` function. We map the `evalf` function to each element of our answer as follows:

```
> map(evalf,");
                              polar(5., .9272952179)
```

Also note that the reverse conversion (from polar form to coordinate form is also possible.

```
> evalc(polar(5, 0.9272952179));

                              3.000000001 + 4.000000000 I
```

8. Inverse Laplace Transforms

A powerful mathematical tool for solving differential equations is the Laplace transform. In this problem, we concentrate on the inverse Laplace transform . We assume that we are given a circuit in the Laplace domain as in Fig. 8.1. Our task is to find the equations for the currents `i1` and `i2` in the time domain.

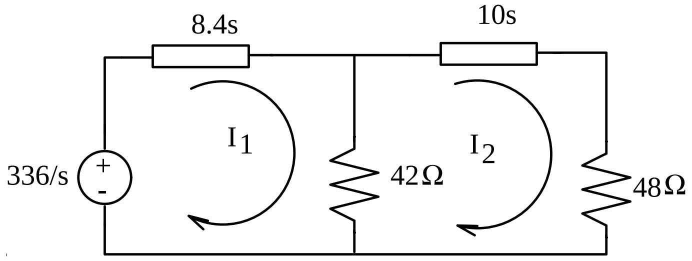

Figure 8.1: The circuit for problem 8.

We begin by writing the two mesh current equations and solving them.

```
> eq1:= 336/s = (42+8.4*s)*i1 - 42*i2;
                                336
                      eq1 := --- = (42 + 8.4 s) i1 - 42 i2
                                 s

> eq2:= 0 = -42*i1 + (90+10*s)*i2;
                        eq2 := 0 = - 42 i1 + (90 + 10 s) i2

> soln:=solve({eq1, eq2}, {i1,i2});
                            168.                              9. + s
      soln := {i2 = --------------------, i1 = 40. --------------------}
                                      2                              2
                    s (24. + 14. s + s )            s (24. + 14. s + s )

> assign(soln);
```

In a paper and pencil approach, our next step would be expansion of these solutions into a sum of partial fractions. However, in *Maple* we can skip this step and directly find the inverse transform. A library call is necessary before we can perform the inverse operation.

```
> readlib(laplace);
proc(expr,t,s) ... end

> evalf(invlaplace(i1, s, t),3);
                   15. - 1. exp(-12. t) - 14. exp(-2. t)

> evalf(invlaplace(i2,s,t), 3);
                  7. + 1.40 exp(-12. t) - 8.40 exp(-2. t)
```

Note that the expression `exp(-12.  t)` is the same as e^{12t}. The period after the number indicates that it is a floating number. Also note the usage of the `evalf` function which limits the number of decimal places to three. This eliminates the redundant zeros for a better visual appeal.

9. Fourier Series

Periodic functions are a source of interest in many engineering problems such as mechanical vibration, fluid flow, heat flow and electric signals. A common mathematical tool for studying periodic functions is the Fourier series. Jean Baptiste Joseph Fourier (1768-1830) discovered that a periodic function can be represented by an infinite sum of sine or cosine functions that are harmonically related. For a periodic function $f(t)$, Fourier showed that $f(t)$ can be expressed as:

$$f(t) = a_v + \sum_{n=1}^{\infty} a_n \cos n\omega_o t + b_n \sin n\omega_o t$$

Consider the waveform shown in figure 9.1. We want to find the Fourier series for this wave form.

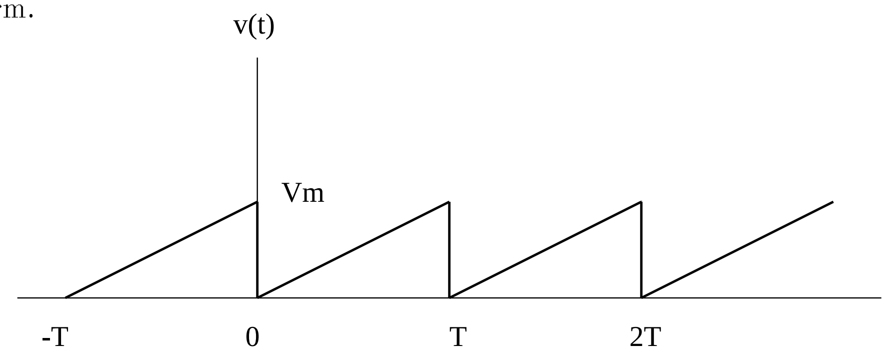

Figure 9.1: The periodic wave for problem 9.

The coefficients a_v, a_n *and* b_n are given by the following relationships:

$$a_v = \frac{1}{T}\int_{t_o}^{t_o+T} f(t)dt,$$

$$a_k = \frac{2}{T}\int_{t_o}^{t_o+T} f(t)\cos k\omega_o t dt,$$

$$b_k = \frac{2}{T}\int_{t_o}^{t_o+T} f(t)\cos k\omega_o t dt.$$

We use the above information along with *Maple*'s calculating engine to derive an expression for the wave shown in figure 9.1 in time domain.

In order to graph our results, we need to assign some numerical values to the variables and thus make them a constant for the particular graph we wish to draw.

```
> w:=2*Pi;
                                w := 2 Pi
> T:=1;
                                 T := 1

> Vm:=5;
                                 Vm := 5
```

The expression for $v(t)$ between 0 and T is defined in *Maple*.

```
> v:=t->(Vm/T)*t;
                                     Vm t
                              v := t -> ----
                                      T
```

Note the use of function notation. We now use the equations shown above to calculate the coefficients.

```
> av:= 1/T * int(v(t), t=0..T);
                                av := 5/2

> ak:= 2/T *int(v(t) * cos(k*w*t), t=0..T);

                   cos(2 k Pi) + 2 k Pi sin(2 k Pi)        5
         ak := 5/2 -------------------------------- - --------
                                2   2                    2   2
                               k  Pi                  2 k  Pi

> bk:= (2/T) * int(v(t) * sin(k*w*t), t=0..T);

                     - sin(2 k Pi) + 2 k Pi cos(2 k Pi)
           bk := - 5/2 ----------------------------------
                                    2   2
                                   k  Pi
```

We now have all the necessary components to express the Fourier series for the waveform. Here we have carried out the sum over three different values for k. We have plotted each sum separately to show how the function expressed by the Fourier series becomes more and more like the original waveform. Theoretically if one carries the summation to infinity, then the exact replicate of the original waveform is obtained.

```
> g:=av + sum(ak * cos(k*w*t) + bk*sin(k*w*t), k=1..5):

> plot(g, t=0..5):

> g:=av + sum(ak * cos(k*w*t) + bk*sin(k*w*t), k=1..15):
```

```
> plot(g, t=0..5):

> g:=av + sum(ak * cos(k*w*t) + bk*sin(k*w*t), k=1..71):

> plot(g, t=0..5):
```

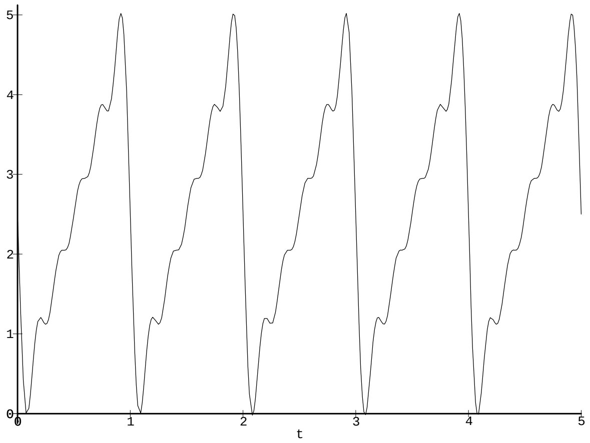

Figure 9.2: Plot of v(t) for k=1..5.

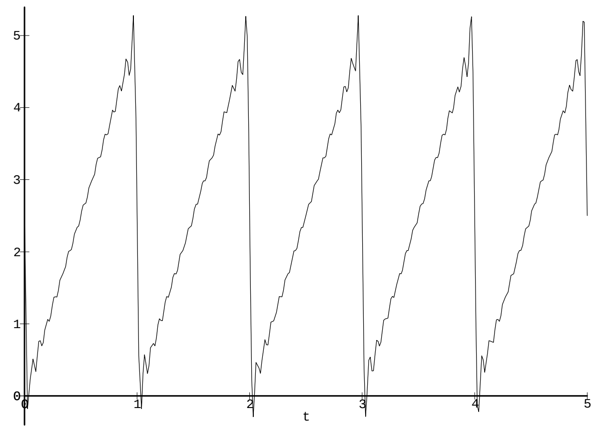

Figure 9.3: Plot of v(t) for k=1..15.

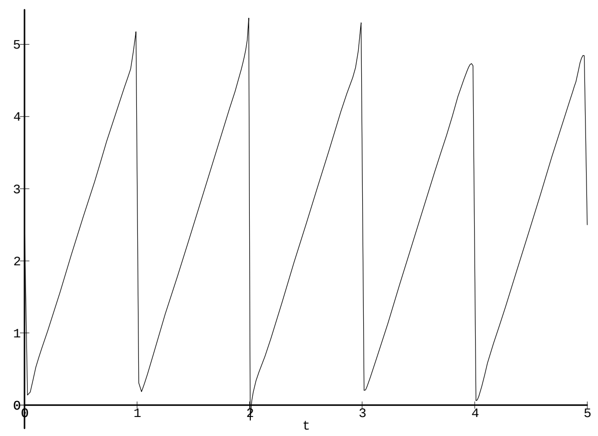

Figure 9.4: Plot of v(t) for k=1..71

10. Vector calculations and the divergence theorem

Vector calculus is an essential part of understanding electromagnetics and mechanics. Vector operations can be tedious and time consuming, especially for complicated vector fields. The following discussion does not supply a specific problem setting, but it shows how several common vector operations are performed in *Maple*.

First we define a vector field **F** in cylindrical coordinates. We define it as a function of variable `z` for reasons explained later.

```
> F:=z->[F1*r, F2*z^2, F3*z];
                                         2
                      F := z -> [F1 r, F2 z , F3 z]
```

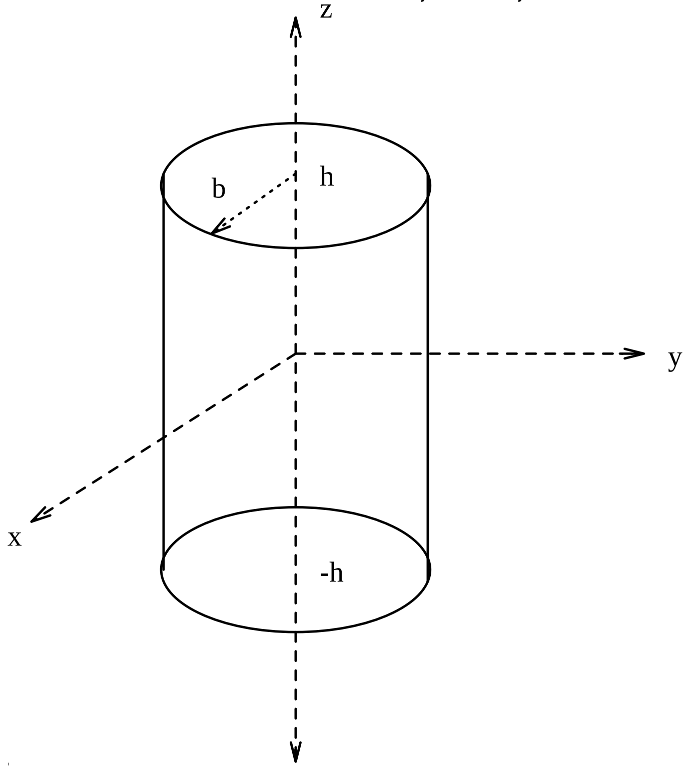

Figure 10.1: Cylinder for Problem 10.

Maple performs vector calculations using the `linalg` package. Actually, vectors are nothing but a one dimensional array as far as *Maple* is concerned. First, let us calculate the total flux of **F** out of the cylinder shown in Fig. 10.1. In mathematical notation, we want to calculate:

$$\oint_S \mathbf{F} \cdot d\mathbf{S}$$

where **S** is the surface of the cylinder(top, bottom and side). The calculation is shown below.

```
> with(linalg):
>

# top surface
> s:=[0,0,r];
                              s := [0, 0, r]

> dotprod(F(h),s);

                                  F3 h r

> top_surf:=int(int(dotprod(F(h),s), r=0..b), phi=0..2*Pi);
                                                2
                          top_surf := F3 h b  Pi

# bottom surface
> s:=[0,0,-r];
                             s := [0, 0, - r]

> bot_surf:=int(int(dotprod(F(-h),s), r=0..b), phi=0..2*Pi);
                                                2
                          bot_surf := F3 h b  Pi

# outer shell
> s:=[b, 0, 0];
                              s := [b, 0, 0]

> outer_surf:=int(int(dotprod(F(z),s), phi=0..2*Pi), z=-h..h);
                        outer_surf := 4 F1 r b Pi h

> total_flux:= top_surf + bot_surf + outer_surf;
                                          2
                total_flux := 2 F3 h b  Pi + 4 F1 r b Pi h
```

Don't be intimidated by the big crowd of equations. It really isn't as difficult as it seems. A cylinder has three surfaces and therefore three integrations are necessary. Note that d**S** is itself a unit vector normal to the surface of integration. This unit vector will be different depending on which surface is being integrated upon. We denote this vector by the letter `s` and as you can see a different vector `s` is defined prior to each integration.

Also, the function `dotprod` is used to perform the dot product calculation required prior to integration. Finally, since double integrals are required to cover a surface, you see an `int` function called within another `int` function.

For the top and bottom surfaces, the value of `z` is fixed and so we use the **F** vector with a fixed value of `h` and `-h` respectively. For the side surface, the variable `z` is no longer fixed (it is one of the integration variables) and thus we use **F**(z).

For clarity, each of the integrations performed above are presented here in mathematical notation. Compare them with the notation used by *Maple* and convince yourself that they are the same. These are the expressions after we perform the dot product calculation.

$$\int_0^{2\pi}\int_0^b F_3 h r dr d\phi$$

$$\int_0^{2\pi}\int_0^b -F_3(-h) r dr d\phi$$

$$\int_{-h}^{h}\int_0^{2\pi} F_1 r r d\phi dz$$

One of the common operations performed on vector fields is the calculation of the divergence of a field. *Maple* has a built-in function for divergence in Cartesian coordinates, but not for other coordinates. It is not very difficult to write your own divergent function for different coordinate systems, however, here we perform the required calculations explicitly. The divergence of a field in cylindrical coordinates is given by the following:

$$\nabla.\mathbf{F} = \frac{1}{r}\frac{\partial}{\partial r}(rF_r) + \frac{\partial F_\phi}{r\partial\phi} + \frac{\partial F_z}{\partial z}$$

For an explanation of how this equation is derived, refer to any vector calculus textbook. Here is how the above equation is done in *Maple*. Note the usage of the `diff` function to differentiate and the `op` function to access the three elements of the vector **F**.

```
> div_F:= 1/r*diff(r*op(1,F(z)),r) + 1/r*diff(op(2,F(z)),phi) + diff(op(3,F(z)),z);
                        div_F := 2 F1 + F3
```

To conclude our discussion, let us calculate the volume integral over the divergence of the field and thus verify the Divergence Theorem. Recall that this theorem states that:

$$\oint_S \mathbf{F} \,.\, d\mathbf{S} = \int_V \nabla \,.\, \mathbf{F} dv$$

We have already calculated the left hand side of this equation. We now calculate the right hand side and see that they are indeed equal. Note that we must evaluate a triple integral for a volume integral.

```
> int(int(int(div_F * r, r=0..b), phi=0..2*Pi), z=-h..h);
                                          2
                          2 (2 F1 + F3) b  Pi h

> expand(");
                             2                    2
                        4 b  Pi h F1 + 2 F3 h b  Pi
```

11. Vector Fields: a graphical approach

A picture is worth a thousand words. When it comes to vectors and the calculus that it accompanies, a graphical intuition or "feel" is a great asset. In this problem, we will explore some of *Maple*'s features which can help us better understand vector calculus.

Let us start by defining two simple vector fields and plotting them.

```
> with(plots):

> with(linalg):
Warning: new definition for   norm
Warning: new definition for   trace

> v1:=[x,y,0];
                                v1 := [x, y, 0]

> fieldplot(v1,x=-5..5, y=-5..5):


> v2:=[y/sqrt(x^2+y^2), -x/sqrt(x^2+y^2),0];
                                 y               x
                   v2 := [------------, - ------------, 0]
                            2    2 1/2       2    2 1/2
                          (x  + y )        (x  + y )


> fieldplot(v2,x=-5..5, y=-5..5):
```

Maple treats vectors as a one-dimensional array. If you come from a background where vector elements were represented by $\hat{\imath}$, $\hat{\jmath}$ *and* $\hat{k}$, then *Maple*'s notation might seem odd at first. They are really the same. For example, $x\hat{\imath} + y\hat{\jmath} + z\hat{k}$ = [x,y,z].

Another point is the use of the function `fieldplot`. It takes the vector field you want to plot along with specified ranges for `x` and `y`, and produces a field plot. There is a similar function `fieldplot3d` which produces a three dimensional field plot.

In the course of vector calculations, one almost always must find either the curl or the divergence of a field. These operations are necessary to take advantage of the Divergence Theorem or Stokes' Theorem. If you are dealing with the Cartesian coordinate system, then simply use the functions provided by *Maple* as follows.

```
> curl(v1,[x,y,z]);
                                  [ 0, 0, 0 ]

> curl(v2,[x,y,z]);
                                             2            2
                           2                x            y
       [ 0, 0, - ------------ + ------------ + ------------ ]
                   2    2 1/2     2    2 3/2     2    2 3/2
                 (x  + y )      (x  + y )      (x  + y )
```

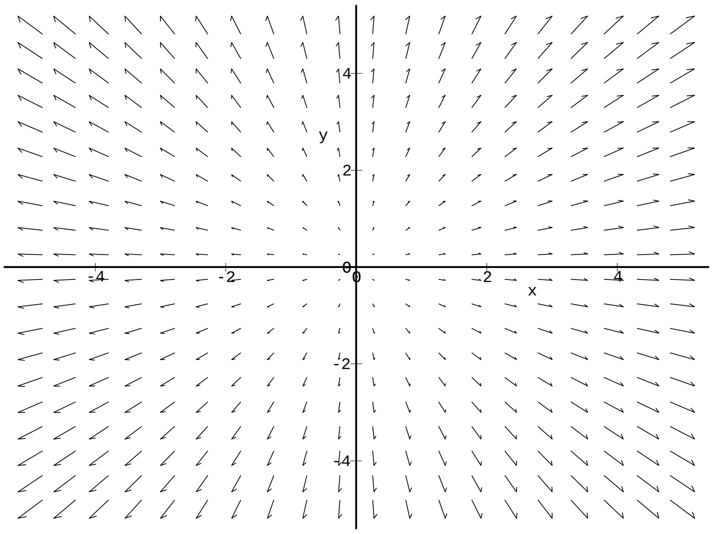

Figure 11.1: Field Plot for **V1**

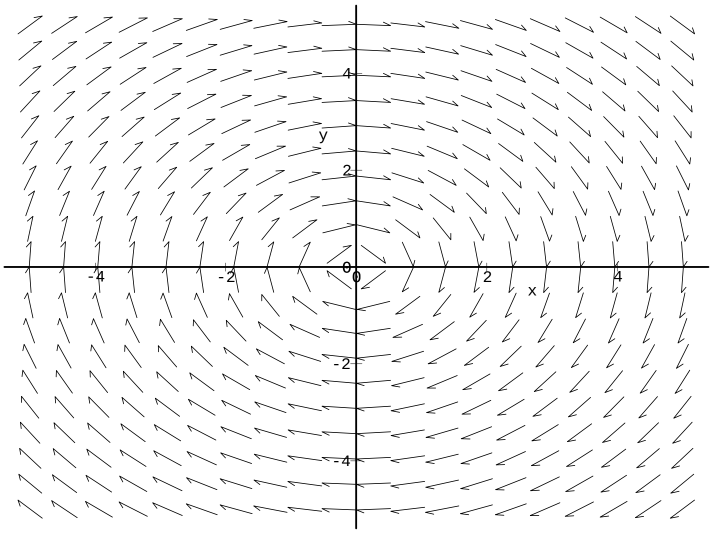

Figure 11.2: Field Plot for **V2**

```
> diverge(v1,[x,y,z]);
                                        2

> diverge(v2,[x,y,z]);
                                        0
```

Aside from the introduction of a few vector related functions, the main purpose of this problem is to emphasize the importance of developing an intuition for mathematical concepts. Upon examination of the first field **V1**, you can see that there is no "twisting" in the field. This should prompt you that the curl of the field is zero, as shown in the above calculation. Similarly, the second field **V2** has no "outward" component. It does not "diverge" from its center, which is a good indicator that the divergence of this field is zero. Try experimenting with different fields and look at the plots and try to make sense out of what you calculate.

12. Boolean Algebra

Boolean Algebra always forms the introductory chapters of Digital Design textbooks. A direct correlation exist between digital components in a circuit and the boolean operators. Well developed methods exist to reduce (simplify) an expression which leads to reduced number of components for a given circuit.

Maple includes a `logic` package which has a number of functions that deal with boolean algebra. Consider the expression represented by Fig. 12.1. Note that we have two OR gates, one AND gate and one INVERTER. We can use *Maple* to simplify the expression as shown in Fig. 12.2. We have reduced the number of gates to one AND gate, one OR gate and one INVERTER.

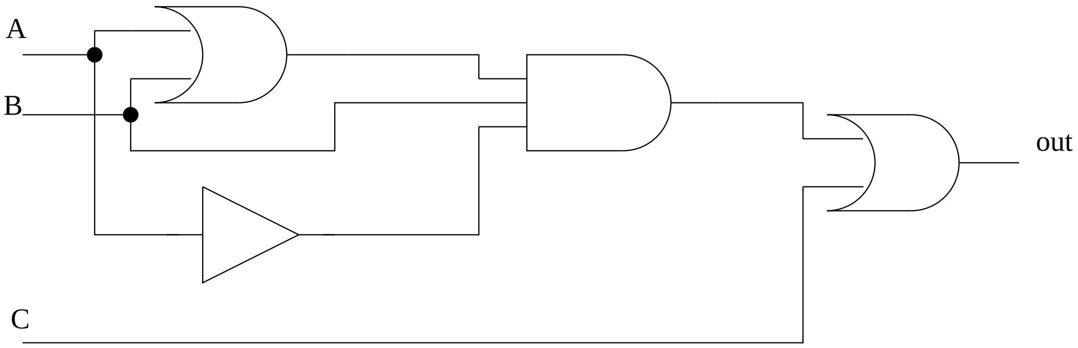

Figure 12.1: Logic Gates for `expr1`

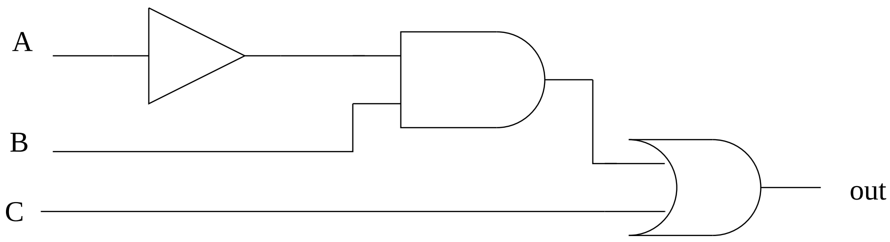

Figure 12.2: Simplified Logic Gates for `expr1`

```
> with(logic):
> expr1:= (a &or b) &and &not a &and b &or c;
              expr1 := (((a &or b) &and (&not a)) &and b) &or c

> bsimp(expr1);
                         ((&not a) &and b) &or c

> convert(bsimp(expr1), frominert);
                              not a and b or c

> convert(bsimp(expr1), MOD2);

                        1 + (1 + (1 + a) b) (1 + c)

> convert(bsimp(expr1), MOD2, 'expanded');
                         c + b + b c + b a + a b c

> bequal(expr1,bsimp(expr1));
                                   true
```

We have used the function `bsimp` to simplify the expression given by `expr1`. *Maple* can display boolean expressions in several formats which are shown above. Refer to the documentation for the function `convert` for a discussion of each format.

Finally, just to convince ourselves that the original expression and the simplified expression are indeed equal, we use the function `bequal` to verify that.

Here is a similar calculation with a more complicated expression.

```
> expr2:= (&not a &or b) &and (c &or a) &and (b &or &not c &or d) &and (b &or d &or e);
expr2 := ((((&not a) &or b) &and (c &or a)) &and ((b &or (&not c)) &or d)) &and

    ((b &or d) &or e)


> bsimp(expr2);
               &or(&and(&not a, c, d), a &and b, c &and b)


> convert(bsimp(expr2), frominert);
                 not a and c and d or a and b or c and b

> convert(bsimp(expr2), MOD2);

                1 + (1 + (1 + a) c d) (1 + b a) (1 + b c)

> convert(bsimp(expr2), MOD2, 'expanded');
              c d + b c + d b c + b a + a c d + a b c + a d b c

> convert(bsimp("), MOD2);
                    1 + (1 + c d) (1 + b a) (1 + b c)

> bequal(expr2, bsimp(expr2));
```

```
                                   true
```

A tautology is a boolean expression which always evaluates to `true` regardless of what the values of its variables are. We can determine if a given expression is a tautology by the `tautology` function.

```
> tautology(expr1);
                                   false

> tautology(a &or 1);
                                   true
```

13. A bit of graph theory

Graph theory is an important tool for analyzing real world problems. It is a modeling tool and once we have a graph model, we can perform a number of operations to enhance our understanding of the system. Computer networks are a good example of where one can use graph theory.

Functions related to graph theory are in a package called **networks**. A graph can be represented by an *adjacency* matrix. Operations on this matrix are ordinary matrix calculations and can yield information about the nature of the graph. Think of the rows and columns of the matrix as the vertices of the graph. A value of '1' indicates that the two vertices are connected and a '0' indicates that they are disconnected.

A theorem states that a graph is connected if every entry in the matrix $\mathbf{A}+\mathbf{A}^2+ ... + \mathbf{A}^{n-1} > 1$, where n is the number of row or columns in the matrix (remember we are dealing with an $n \times n$ matrix). We use the facilities provided by the **linalg** package to test this theorem for a given matrix **A**.

```
> with(networks):

> with(linalg):
Warning: new definition for   charpoly
Warning: new definition for   norm
Warning: new definition for   rank
Warning: new definition for   trace

> A:=matrix(5,5,[0, 0, 0, 1, 1, 0, 0 ,1 ,0, 0, 0,1,0,0,0, 1,0,0,0,1, 1, 0, 0,1,0]);
                                  [ 0  0  0  1  1 ]
                                  [               ]
                                  [ 0  0  1  0  0 ]
                                  [               ]
                             A := [ 0  1  0  0  0 ]
                                  [               ]
                                  [ 1  0  0  0  1 ]
                                  [               ]
                                  [ 1  0  0  1  0 ]

> evalm(A+A^2+A^3+A^4);
                              [ 10  0  0  10  10 ]
                              [                  ]
                              [  0  2  2   0   0 ]
                              [                  ]
                              [  0  2  2   0   0 ]
                              [                  ]
                              [ 10  0  0  10  10 ]
                              [                  ]
                              [ 10  0  0  10  10 ]
```

The result shows that this graph is not connected, since there are some elements which are not greater than 1 (e.g., A[2,1] = 0).

Now let us draw the graph and examine connectivity from a pictorial view.

```
> new(G):
> addvertex(v1,v2,v3,v4,v5,G);
                                    v2, v1, v3, v4, v5
> addedge([{v1,v4}, {v1,v5}, {v4,v5}, {v2,v3}],G);
                                      e1, e2, e3, e4
> draw(G):
```

A new graph is defined via the **new** function. This defines an empty graph. We use the **addvertex** function to indicate the vertices of our graph. We then use the **addedge** function to draw the connections between the vertices. The picture of the graph is shown in Fig. 13.1.

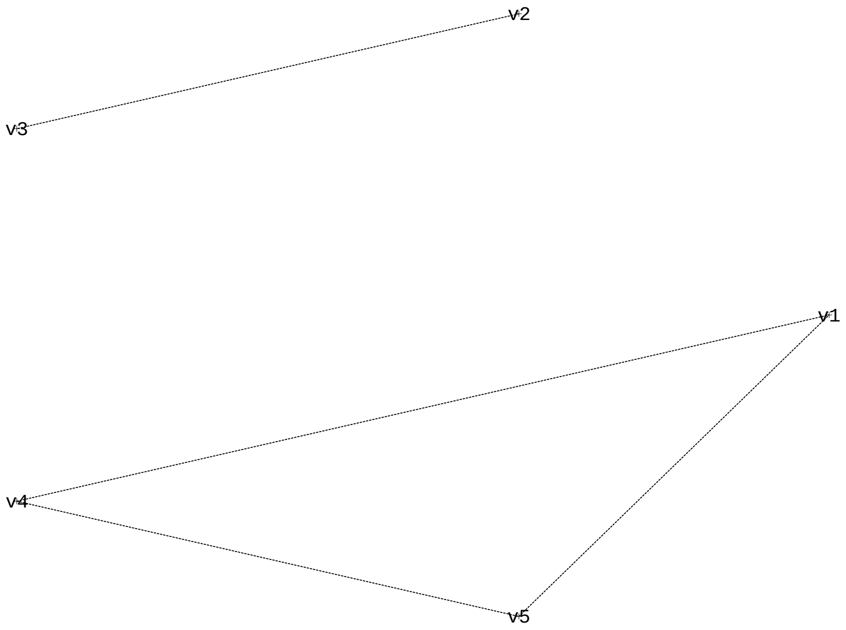

Figure 13.1: A disconnected graph described by the matrix **A**.

Note that there is no connection between vertices 2 and 5. Thus this graph is disconnected. We can have *Maple* check for connectivity via the built-in function `connectivity`.

```
> connectivity(G);
```

```
                                        0
```

Finally, the `adjacency` function takes a graph and arrives at the adjacency matrix for that graph. In our case, we started with the matrix, but this function can be useful if the operation you want to perform on the graph is not provided by *Maple*, but you have a algorithm that can be applied to the adjacency matrix.

```
> adjacency(G);
                              [ 0  0  0  1  1 ]
                              [               ]
                              [ 0  0  1  0  0 ]
                              [               ]
                              [ 0  1  0  0  0 ]
                              [               ]
                              [ 1  0  0  0  1 ]
                              [               ]
                              [ 1  0  0  1  0 ]
```

This concludes our brief introduction to graph theory.

14. Eigenvalues and eigenvector

The ability to determine eigenvectors and eigenvalues is essential when it comes to solving systems of differential equations. Recall that the eigenvalues λ and eigenvectors $\mathbf{x}$ satisfy the equation $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$.

Let's start with a 3×3 matrix, $\mathbf{A}$. Maple has an entire package dedicated to matrices and matrix operations. It is called `linalg` which is an abbreviated form of linear algebra. We use the `matrix` function to create our matrix.

```
> with(linalg):
```

```
> A:=matrix(3,3,[0,1,1,1,0,1,1,1,0]);
                                        [ 0  1  1 ]
                                        [         ]
                                   A := [ 1  0  1 ]
                                        [         ]
                                        [ 1  1  0 ]
```

Actually, *Maple* treats a matrix as a two-dimensional array and therefore individual matrix elements can be accessed via array subscripts (e.g., A[2,2]).

One call to the function `eigenvals` is all that is needed to find the eigenvalues for our matrix.

```
> eigenvals(A);
                                      2, -1, -1
```

If you are interested in the eigenvectors, then the preferred method is a call to the function `eigenvects` as follows:

```
> vl:=eigenvects(A);
      vl := [2, 1, {[ 1, 1, 1 ]}], [-1, 2, {[ -1, 1, 0 ], [ -1, 0, 1 ]}]
```

According to the help page for the `eigenvects` function, the result returned is a sequence of lists of the form [e_i, m_i, v[1,i],...v[n_i,i]], where the e_i are the eigenvalues, m_i their algebraic multiplicities and v[1,i], ..., v[n_i,i] is the set of basis vectors for the eigen-space corresponding to e_i, and 1 <= n_i <= m_i is the dimension of the eigenspace. Basically, you get everything with one shot.

To test the result and also to get some practice in matrix calculations we will verify the equation $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$.

In order to perform the calculation required we need an identity matrix and we generate it with a special call to the `array` function.

```
> Id:= array(identity, 1..3,1..3);
                    Id := array(identity, 1 .. 3, 1 .. 3, [])
```

```
> print(Id);
```

```
[ 1  0  0 ]
[         ]
[ 0  1  0 ]
[         ]
[ 0  0  1 ]
```

We now have all the components necessary to perform our calculation. Since we are dealing with matrices, we must tell *Maple* to treat the calculation as a matrix calculation. A simple way to do this is by using the `evalm` function, which evaluates an expression using "matrix" arithmetic.

```
> evalm( (A-2*Id)*[1,1,1]);
                                        [ 0, 0, 0 ]

> evalm( (A-(-1)*Id)*[-1,1,0]);
                                        [ 0, 0, 0 ]

> evalm( (A-(-1)*Id)*[-1,0,1]);
                                        [ 0, 0, 0 ]
```

The evaluation method shown above works fine for the small matrix **A**. Note that we took the information provided to us by the `eigenvects` function and manually entered them to evaluate the expression. This would not be convenient if **A** was a larger matrix, thus having large eigenvectors. We can perform the same evaluation using the `op` function. For the first eigenvector we use the first element of the array as follows:

```
>vl[1];
                               [2, 1, {[ 1, 1, 1 ]}]

> evalm( (A-op(1,vl[1])*Id)* op(1,op(3,vl[1])));
                                        [ 0, 0, 0 ]
```

On a first look, it might seem strange as to why we have used the `op` function twice to get the vector [1,1,1]. The reason is that the third element of the list stored in vl[1] is actually a set with one member, namely the vector [1,1,1].

```
> op(3,vl[1]);
                                      {[ 1, 1, 1 ]}
```

So in order to bypass the set and get to the sole member of the set we must apply the `op` function a second time as shown in the example above.

Working with matrices can be somewhat tricky with *Maple*, but the powerful tools certainly make up for that. The `linalg` package provides a great number of tools needed to work with matrices. If you plan to do a lot of matrix calculations, familiarize yourself with this package.

15. Fourier Transforms

The Fourier transform is useful in certain communications theory and signal processing situations. Without going into the details of this transform, we present a problem and use Fourier transform to solve it. Our goal is simply to familiarize the reader with using this transformation under *Maple*.

Consider the circuit shown in figure 15.1. The expression for the driving current is given as:

$$i_g(t) = 50cos3tA$$

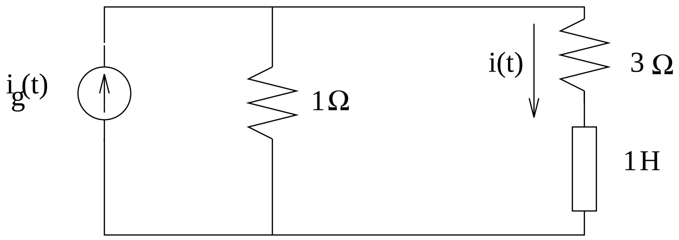

Figure 15.1: The Circuit for problem 15.

We want to use the Fourier transform method to find $i(t)$. Use the following command to load the Fourier transform functions.

```
> readlib(fourier);
proc(F,t,w) ... end
```

We now define the driving current and find its Fourier transform.

```
> igt:=50*cos(3*t);
                              igt := 50 cos(3 t)

> igf:=fourier(ig, t,w);
                    igf := 50 Pi (Dirac(w - 3) + Dirac(w + 3))
```

We can use the transfer function to find an expression for $i(t)$ in frequency domain.

```
> H:=1/(4+w*I);
                                         1
                                 H := -------
                                      4 + I w

> iof:=igf*H;
```

```
                 Pi (Dirac(w - 3) + Dirac(w + 3))
       iof := 50 --------------------------------
                              4 + I w
```

One call to `invfourier` will give us an expression for $i(t)$ in time domain. Note the order of the parameters `w` and `t` are opposite to the call to the `fourier` function.

```
> iot:=invfourier(iof, w,t);
                          4 - 3 I       4 + 3 I
               it := ------------ + ----------
                      exp(- 3 I t)   exp(3 I t)

> simplify(evalc(combine(iot)));
                      8 cos(3 t) + 6 sin(3 t)
```

16. Electromagnetic Field Modeling using *Maple*

Introduction

Introductory courses for electromagnetics are perceived as one of the most difficult classes by the majority of undergraduate students. One reason is a heavy emphasis on mathematics, particularly vector calculus. The second reason is a lack of an intuition about fields and electromagnetics.

A computer algebra system such as Maple can ease this fear by allowing students to manipulate mathematical formulas and interpret the results graphically, thus allowing them to develop an intuition about electromagnetic fields in an experimental fashion.

The following article describes a simplified electromagnetic problem and how Maple is used to model it, solve it and analyze the solution. The approach taken can be used to model more complicated problems. The assumption is made that the reader has some background in vector calculus. For a derivation of the formulas used, the reader is encouraged to consult an introductory text in electromagnetics.

Problem Definition

The problem involves a branch of electromagnetics called electrostatics. In electrostatics the electric charges are assumed to be stationary, thus simplifying the equations describing the fields. Figure 16.1 shows the setting of the problem. There is a large, rotating sphere in the center. Two smaller spheres are connected to the larger sphere via a thin copper band. The angle between the two rods is specified by θ and is fixed at $\pi/6$ for this problem. The angle specified by ϕ varies as the large sphere rotates on a fixed axis.

The problem asks for preparation of an analysis which:

1. Identifies the position of the spheres that causes the highest electric field intensity at the position of the test equipment;

2. Explains how the charge is distributed between the conducting spheres; and

3. Includes a clear contour plot of the voltage field set up by the three charged spheres when in the position identified in (1).

Charge Distribution

Before we can determine the electric field intensity, we must find the charge distribution among the spheres. Intuitively, we can predict that the majority of the charge will reside on the large sphere and that the smaller spheres will have an equal amount of charge. Let Q denote the total charge which is 100nC. Let Q_1, Q_2, Q_3 denote the

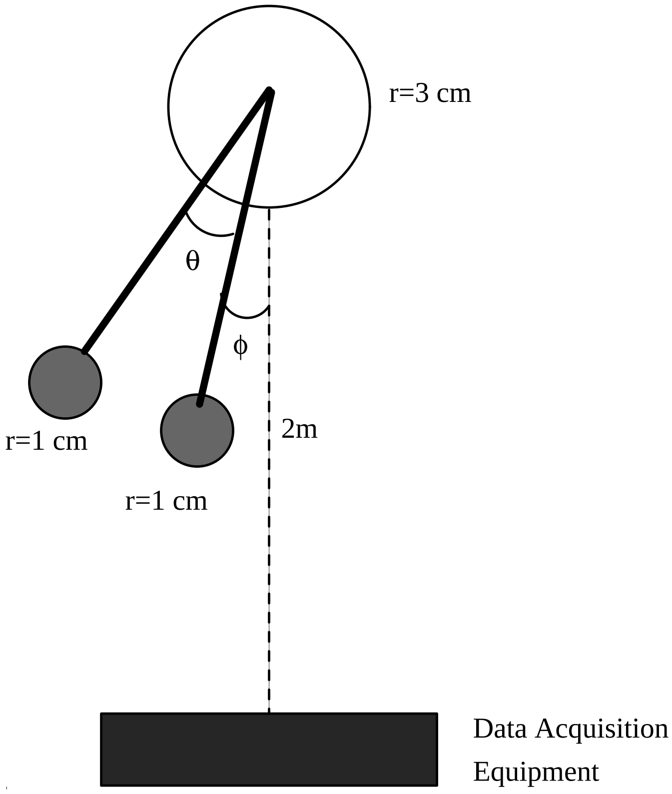

Figure 16.1: Physical setting of the problem

charge distribution on each sphere where Q_1 is the charge on the large sphere. Also let b_1,b_2 and b_3 denote the radii of the spheres where b_1 is the radius of the large sphere. The charges on the spheres are directly proportional to their radii (for a proof refer to an electromagnetic textbook). Therefore, we arrive at the following equations:

$$Q_1 = \frac{b_1}{b_1 + b_2 + b_3}$$

$$Q_2 = \frac{b_2}{b_1 + b_2 + b_3}$$

$$Q_3 = \frac{b_3}{b_1 + b_2 + b_3}$$

We know that $b_1 = 0.03m$ and $b_2 = b_3 = 0.01m$. Therefore, $Q_1 = 60nC$ and $Q_2 = Q_3 = 20nC$.

Electric Field Intensity

We now have three spheres with a known amount of charge on each. We can use vector analysis to derive an expression for the electric field intensity in terms of the rotational angle, ϕ. The electric field intensity at an arbitrary point is given by:

$$\mathbf{E} = \frac{q(\mathbf{R} - \mathbf{R}')}{4\pi\epsilon_o|\mathbf{R} - \mathbf{R}'|^3}.$$

Note that $\mathbf{E}$, $\mathbf{R}$ and $\mathbf{R}'$ are vectors. $\mathbf{R}$ locates the point where we want to measure the electric field and $\mathbf{R}'$ locates the source of the electric field (in our case a charge density).

Due to the symmetry of the sphere we may assume that the electric field outside of the sphere is the same as though the total charge is concentrated on a single point charge at the center. Thus, our problem can be simplified to three point charges in space.

We define this function along with a number of constants as follows:

```
> with(linalg):
> with(plots):
> Epsilon := 10^(-9)/(36*Pi):
> q1:=60*10^(-9):  q2:=20*10^(-9): q3:=20*10^(-9):
```

Note that the denominator of (16) requires the calculation of the magnitude of the difference of vectors $\mathbf{R}$ and $\mathbf{R}'$. We therefore write a procedure (you can think of it as a callable function) which calculates the magnitude of a vector and returns this value. Here is the Maple code.

```
> magnitude := proc(V) RETURN(sqrt(V[1]^2+V[2]^2)) end:
```

We are now ready to define (16) in Maple.

```
> Ef:=(q,R,Rp)->(evalm(q*(R-Rp)/(4*Pi*Epsilon * (magnitude(evalm(R-Rp)))^3)));

                                                 q (R - Rp)
      Ef := (q,R,Rp) -> evalm(1/4 -----------------------------------)
                                                                    3
                                  Pi Epsilon magnitude(evalm(R - Rp))
```

The above functions will be the driving engine of our calculations. Take a moment to familiarize yourself with the definition and compare it to (16).

We will derive the expression for the electric field intensity due to each point charge. The magnitude of the sum of these vectors, will be a function of ϕ, which we can then find its maximum. Before proceeding further, we should define a geometric reference. The center of the large sphere will be the origin and we will utilize the Cartesian coordinate system.

1. Big sphere

```
> R1:=vector([0, -2]):
> Rp1 := vector([0,0]):

> E1:=Ef(q1, R1, Rp1);
                               E1 := [ 0, -135 ]
```

2. Small sphere (separated by angle ϕ from the vertical dashed line)

```
> R2:= vector([0, -2]):
> Rp2:=phi->vector([-sin(phi), -cos(phi)]):
> E2:=Ef(q2, R2, Rp2(phi)):
```

3. Small sphere (separated by angle $\phi + \theta$ from the vertical dashed line)

```
> R3:= vector([0, -2]):
> Rp3:=phi->vector([-sin(phi+Pi/6), -cos(phi+Pi/6)]):
> E3:=Ef(q3,R3, Rp3(phi)):
```

We now have an expression for the electric field intensity at the point of interest due to each sphere. Note that these are vector fields. Also, note that the above expressions are all a function of ϕ. Our next step is a numerical analysis problem in which we sum these vector fields and find the magnitude of their sum. We will then maximize the function in terms of the angle ϕ.

The `evalm` function which is part of the the `linalg` package can be used to perform vector/matrix arithmetic. We use it to sum the three vectors described above and find their magnitude:

```
> etot:=magnitude(evalm(E1+E2+E3)):
```

Note that `etot` is not a vector expression. It is a simple expression in terms of the angle ϕ which can be plotted and maximized. We now plot `etot` to get an intuition of what is going on. Figure 16.2 shows this plot.

```
> with(plots):
> plot(etot, phi=-Pi..Pi);
```

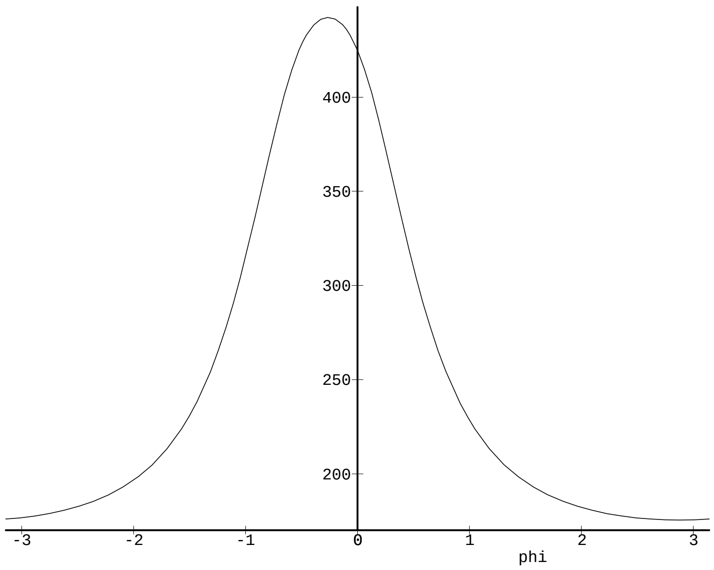

Figure 16.2: Electric Field Intensity as a Function of ϕ

It seems from the graph that the function reaches its peak value at approximately $-15°$. We know that a function has its maximum or minimum when its first order derivative is equal to zero. Using this fact, we can calculate the exact value of the angle ϕ.

```
> fsolve(diff(etot, phi), phi, -.3..0);
                                    -.2617993878

> evalf(-.261799*180/Pi);
                                    -14.99997778
```

Here we have instructed *Maple* to symbolically differentiate the expression `etot` with respect to the variable `phi`. This derivative is then solved in terms of `phi` to give us the angle in radians. We have specified a range from -0.3 to 0 (`-.3..0`) to help Maple in finding the solution. These numbers are approximations from the plot.

The result makes sense because at this position the small spheres are at their symmetrically closest point to the device, contributing the most to the field intensity at the data acquisition equipment. This concludes our analysis of the first part of the problem.

Voltage Field

We now examine the voltage field set up by the three charged spheres when in the position identified in the previous section (i.e., $\phi = -14.99997778$).

The electric potential of a point at $\mathbf{R}$ due to a system of n discrete point charges q_1, q_2, ..., q_n located at $\mathbf{R'_1}$, $\mathbf{R'_2}$, ..., $\mathbf{R'_n}$ is, by superposition, the sum of the potentials due to the individual charges:

$$V = \frac{1}{4\pi\epsilon_o}\sum_{k=1}^{n}\frac{q_k}{|\mathbf{R}-\mathbf{R'_k}|}.$$

We now calculate the voltage field due to each sphere. Note that $\mathbf{R}$ is a vector pointing to where we are interested in the voltage field (i.e. everywhere) and is the same for all three spheres.

```
> Vf:=(q,R,Rp)->q/(4*Pi*Epsilon * magnitude(evalm(R-Rp)));

                                                q
          Vf := (q,R,Rp) -> 1/4 -----------------------------------
                                Pi Epsilon magnitude(evalm(R - Rp))

> R:= vector([x, y]):
```

$\mathbf{R}'$ points to where the charge is and will be different for each sphere. Once again we will treat the spheres as point charges. Also, note that the origin is the center of the big sphere.

```
> Rp1:= vector([0,0]):
> Rp2:= phi->vector([-sin(phi), -cos(phi)]):
> Rp3:=phi->vector([sin(phi), -cos(phi)]):


> V1:= Vf(q1, R, Rp1);
                                        540
                              V1 := ------------
                                       2    2 1/2
                                     (x  + y )

> V2:=Vf(q2, R, Rp2(-0.261799));
                                        180
                V2 := -------------------------------------------
                                        2                     2 1/2
                      ((x - .2588186705)  + (y + .9659259267) )

> V3:=Vf(q3, R, Rp3(-0.261799));
                                        180
                V3 := -------------------------------------------
                                        2                     2 1/2
                      ((x + .2588186705)  + (y + .9659259267) )
```

We can use the `plots` package to draw a contour plot of the voltage field. The result is shown in 16.3.

```
> contourplot(V1+V2+V3, x=-1..1, y=-2..2, grid=[40,40]);
```

We can see that near the center of each sphere the voltage field is changing fast, indicating strong electric field intensity. As we get further away from the system, the voltage lines become more and more separated until our system of three point charges appears as a central point charge (i.e., the voltage line is a circle around all three point charges).

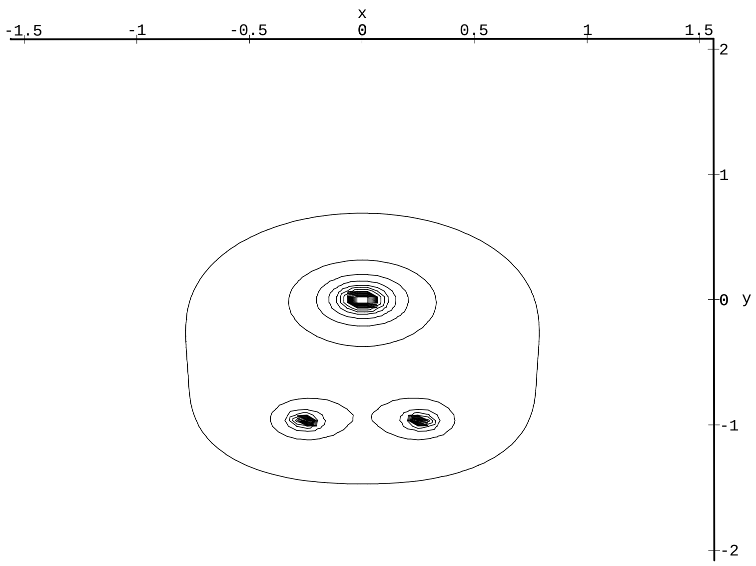

Figure 16.3: Voltage Field Contour Plot

The reader is encouraged to experiment with different variations for the x and y ranges and the `grid` parameter to obtain more details about the voltage field.

Conclusion

We have provided both an analytical and numerical analysis of the three-part problem. We made certain assumptions to simplify the problem and took advantage of the symmetry involved. Our results fit our intuitive expectations. Hopefully, students will find this "self-discovery" approach to problems more appealing than the usual "cookbook" method. Maple is the perfect tool for developing one's intuition about science and engineering.

References

[1] Boyce, William & DiPrima, Richard, *Elementary Differential Equations and Boundary Value Problems*, John Wiley & Sons, 1992.

[2] Dierker, Paul & Voxman, William, *Discrete Mathematics*, Harcourt Brace Jovanovich, 1986.

[3] Cheng, David, *Field and Wave Electromagnetics*, Addison-Wesley, 1990.

[4] Swokowski, Earl, *Calculus with Analytic Geometry*, PWS-KENT, 1988.

[5] Nilsson, James, *Electric Circuits*, Addison-Wesley, 1993.

[6] Benson, Harris, *University Physics*, John Wiley & Sons, 1991.

Index