

Chapter 1

Systems

1.1 On Line

In this introductory section we will pose no exercises, but instead, will detail how to use Maple to solve problems in linear algebra. For the novice Maple user, this section is essential reading and reference. For the experienced Maple user, this section can be examined for evidence of any new tips and ideas for working with Maple. In addition, this section will reveal the working style guiding the solutions in the rest of this manual.

Maple is a symbolic, as well as a numeric, language. For users with experience in numerical computation (Matlab, BASIC, FORTRAN, C, etc.) there are differences in thinking that accompany calculating in Maple. The presence of symbolic variables adds a dimension missing in strictly numeric languages. For one thing, Maple is capable of computing answers in exact (symbolic) form, so round off error need not be an issue in didactic experiments. For another, it really helps to have a vision as to how to mix symbolic and numeric calculations, lest operations performed numerically impinge on the operations which will be later performed symbolically.

We illustrate some sample Maple calculations, beginning with simple arithmetic. As we present the Maple syntax for various calculations, we will explain the Maple Release 4 interface as it appears under Windows on a PC. Release 4 has been engineered to have the same functionality on all major platforms (Macintosh, PC, UNIX) so only local differences in file structures and networks might possibly differ.

Enter Maple syntax at the prompt, the ">" at the left edge of the screen.

```
> 2/3 + 5/7;
```

Lines entered into Maple must have terminal punctuation, typically, the semi-colon (;). Simple arithmetic is done automatically. A new prompt is generated upon execution of the command, and that follows upon pressing the ENTER key. Spaces are generally ignored by Maple, and are used here to improve readability.

```
> x + 2*x;
```

Maple applies the rules of algebra to symbolic expressions, and some of the simplifications are immediate. Others must be requested by a variety of special commands.

Assignments are made using the two characters "colon" and "equal" (:=), two characters that must not be separated by a space!

```
> f := x^2;
```

We have just entered an *expression*, or formula, and assigned it to the name "f." To evaluate this expression at $x = 3$, use the **subs** command to substitute $x = 3$ into f.

```
> subs(x=3, f);
```

We next articulate Lopez's Large Law (see the article Tips for Maple Instructors, *MapleTech*, Vol. 3, NO. 2, 1996, published by *Birkhauser*) which states "never use on the left of an assignment a variable in use somewhere on the right." Thus, it is not a good idea to make the assignment $x := x + 1$. Although this is an extremely common construction in numeric languages, it is most unwise in a symbolic language.

Moreover, Lopez's Large Law precludes assigning values to the "working variables" x, y, z, etc. A strict distinction between the letters being used for "names" on the left, and "variables" on the right makes Maple more user-friendly.

Since we have demonstrated how to make an assignment to the name "f" we next show how to "erase" that assignment.

```
> f;
> f := 'f';
> f;
```

First, "erasing" consists of assigning f to its "letter value" which is what the single quotes accomplish. Second, entering "f;" interrogates Maple for what it knows about that name. Maple echoes the contents of that name.

It is possible to create unfathomable Maple Worksheets. Violations of Lopez's Large Law can lead to such difficulties, and so can misunderstand-

ing Maple's file management. For example, suppose Lopez's Large Law has been violated by

```
> x := 3;
```

Much later in the Worksheet, the "variable" x is used symbolically, as in

```
> f := x*sin(Pi*x);
```

What happened? Why didn't the formula $x \sin \pi x$ appear? Since x has been assigned the value 3, Maple computed $\sin(3\pi)$ and got 0.

```
> x := 'x';
```

```
> f;
```

Violating Lopez's Large Law indeed has consequences. It is not enough to "erase" x. Once the assignment to f is the number 0, it remains the number 0. Having a clear strategy for working with Maple prevents needless frustration.

The next "time-bomb" is more subtle. Suppose that f has the value 0 from the above calculations. And suppose that all record of the existence of f in the Worksheet is removed by deleting the appearance of f from the Worksheet. The letter "f" is still assigned the value 0. In fact, it's worse than that. If another blank Worksheet is opened via the menu options File/New, this new Worksheet will still have f assigned the value 0. This is because both Worksheets share the same memory, and what is known to Maple in one Worksheet is known to the other. On some platforms, with the right initialization of Maple, each Worksheet can have its own attached memory. Unless you know for sure how your copy of Maple is installed, it is best to assume that all Worksheets share a single memory, and always, under all circumstances, remember that merely removing the appearance of an assignment from a Worksheet does not remove that assignment from Maple's memory.

Suppose at this point in your experiments with a Worksheet you have excised the last input/output pair, and the bottom of your Worksheet no longer contains "the next prompt." How do you generate a new prompt? Place the cursor in an input line. Simultaneously press the keys CTRL and k to insert a prompt *above* the cursor, and CTRL and j to insert a new prompt *below* the cursor. If you examine the Insert menu, the "Execution Group" corresponds to "a new prompt."

Having toyed with the menu bar, observe the Help menu. The best advice a Maple user can receive is to begin with Help/Contents. The resulting document that opens is hyperlinked to all sorts of information about Maple,

and it is left to the user to navigate through instructions on the interface, and on Maple itself.

However, to get help on a command whose name you know, you can use the question mark.

```
> ?subs
```

Help commands don't need terminal punctuation. At the bottom of most help screens is a section of Examples. Look there first. Examples can be copied and pasted back into your Worksheet for experimentation.

The exercises for each section in this manual begin with the instruction to "restart Maple." This means to issue the **restart** command which clears all variables. This command does not erase anything visible in the Worksheet, so if a Worksheet has become confused and entangled, issuing a restart and then working from the top down re-executing all the entered commands, is a way of "beginning at the beginning" and retracing the thought process in the Worksheet.

```
> restart;
> f;
```

We next illustrate some algebraic simplifications.

```
> q := 1/x + 1/y;
> q1 := simplify(q);
```

Commands in Maple typically take parentheses around the argument or arguments. The letter "q" makes a handy label because it is easy to find on the keyboard. Re-assigning a new version of a name to itself is actually a violation Lopez's Large Law, and should be avoided. Thus,

```
> q := simplify(q);
```

is not a syntax error, but is just plain bad workmanship. If parts of the Worksheet are re-executed, which version of q is being referenced? Is it the unsimplified version or the simplified version? So, use unique names for each meaningful Maple output to avoid confusion when experimenting, since that usually requires editing, changing, re-executing, moving up and down throughout the Worksheet. If the same letter has multiple meanings throughout the Worksheet, chaos results.

Maple crashes. It is a fact of life that Maple, inexplicably and explicable, crashes. It is exceedingly frustrating to have spent an hour or more on an assignment in Maple, only to lose it all because Maple crashed. There is only one piece of advice that makes sense here, and that is "Save early, and save often." The first time a Worksheet is saved (File/Save or the "diskette"

icon on the toolbar) Maple prompts for a file name and a destination for the saved file. Thereafter, clicking the "diskette" icon on the toolbar, or entering CONTROL S from the keyboard, saves work to that same file. Save early and save often. That advice cannot be repeated too frequently.

Maple is both a symbolic and a numeric language. Thus, there is a difference between 1 and 1.0 in Maple. The first is the exact integer 1, while the second is the decimal version of the number 1. Converting the exact symbolic representation of a number is done with the **evalf** (evaluate floating point) command.

```
> q := 1/sqrt(2);
> evalf(q);
```

Note that Maple immediately rationalizes $\frac{1}{\sqrt{2}}$. And note further that Maple can provide many more than the default 10 digits.

```
> evalf(q,20);
```

The next thing useful to know about Maple is how to reference parts of answers it generates. Consider the following solution to a quadratic equation.

```
> q := x^2 + 3*x + 1 = 0;
> solve(q,x);
```

Maple has returned a *sequence* of two roots, which can be referenced if a tag had been assigned to the **solve** command. Thus, the better working strategy is

```
> q1 := solve(q,x);
```

Now, the roots can be referenced by the bracket notation

```
> q1[1];
> q1[2];
```

Thus, there are three data structures Maple uses that are worth understanding. Maple uses *sequences*, *lists*, and *sets*. In a sequence items are separated by commas. A list is a sequence enclosed by square brackets: [a,b,c]. A set is a sequence enclosed by curly braces: {a,b,c}. The list preserves order and replicas. The set does not.

```
> [a,b,a,a,c];
> {a,b,a,a,c};
```

Each structure reflects valid mathematical usage, and Maple has commands for manipulating each data structure properly.

Although repetitive tasks can be implemented in Maple by copying and pasting input lines, a for-loop is the appropriate way to repeat similar instructions. In this manual, the for-loop is entered as a single input as follows.

```
> for k from 1 to 3 do x.k := k^2; od;
```

All three input lines are connected to the one prompt by entering all lines but the last with SHIFT ENTER, rather than simply ENTER. In Release 4 this is now more aesthetic than practical, but if the three lines of code above are entered into Maple V Release 4 with just the ENTER key, the first line will generate a complaint about "incomplete", a complaint that disappears when the terminating `od` ("do" spelled backwards) is entered. In previous versions of Maple, failure to keep the lines of a for-loop together could lead to terrible consequences if changes were made to individual lines of the loop. In Release 4 this is no longer such a problem.

One advantage of the notation `x1`, `x2`, `x3` is that such objects can be referenced collectively by

```
> x.(1..3);
```

For large collections of similar objects this turns out to be a handy device for saving repetitive and tedious typing.

We will be concerned primarily with Maple's functionality in linear algebra. Maple's code is modularized, bundled into related groups called packages. There are some 32 packages in Release 4, all of which are present in every properly installed version of Maple. The command

```
> ?packages
```

brings up the list of packages, and the names of the packages are hyper-linked to more information about the individual packages.

The package we will use most is the *linalg* package, itself containing more than 100 commands for manipulating vectors and matrices. The *linalg* package is "loaded" into Maple via the command

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

Notice that the terminal punctuation here is the colon (`:`) which suppresses output. This package will be loaded for every exercise set, and we will want to suppress the listing of the more than 100 commands made present by this package.

First, we enter the matrix $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

```
> A := matrix(2,2,[1,2,3,4]);
```

It appears easier to provide the **matrix** command with a single list of entries, letting Maple wrap them according to the dimensions given first. The alternative is to give the **matrix** command a list of lists, the sub-lists being the rows of the matrix. Thus,

```
> matrix([[1,2],[3,4]]);
```

We will use Maple's **randmatrix** command to generate matrices at random. Maple's random number generator will generate the same sequence of random numbers each time it is initialized by starting (or restarting) Maple. The advantage here is that results are then reproducible, even if random matrices have been used. There is a way of setting the starting point for the random number generator, but that is not used in any of the exercises. The help file for **rand**, the random number generator, accessed by `?rand`, will mention the global variable `_seed` which can be assigned values (a student's SSN?), thereby making unique assignments for each student. That is not done in these exercises.

```
> B := randmatrix(2,2);
```

Matrix and vector arithmetic is most easily done by applying **evalm** (evaluate matrix) to the desired arithmetical commands.

```
> 2*A + 3*B - A^2;
```

```
> evalm(2*A + 3*B - A^2);
```

In the first instance, merely the names are manipulated by Maple. In the second, the actual entries of the matrices are manipulated. There are times and places for each approach, but only the second is used in these exercises.

Next, we address matrix multiplication, a process that is known in mathematics to be noncommutative. Thus, for numbers, $2*3 = 6$, but for matrices, $A B$ rarely equals $B A$. Hence, Maple distinguishes between the use of `*` for commutative multiplication, and `&*` for the noncommutative multiplication of matrices. In Release 4 Maple will warn specifically that $A*B$ for matrices must be changed to $A \&* B$. In earlier versions, the user had to be prescient.

```
> evalm(A * B);
```

```
> evalm(A &* B); evalm(B &* A);
```

Surprisingly, vectors will take more discussion than matrices. First of all, enter the vector $\mathbf{V} = \begin{bmatrix} -2 \\ 5 \end{bmatrix}$ with the following syntax. In a newly

installed copy of Maple V Release 4 the output will be as you see below.

```
> V := vector([-2,5]);
```

Throughout this manual the output will instead appear as

```
> V := vector([-2,5]);
```

Why the difference and how do we get Maple to render vectors as "column-like" rather than "row-like"? And are such vectors "column vectors" or "row vectors"?

First, no matter how we get Maple to print the vector, it is always a column vector.

Second, to get your Maple session to print the vectors as columns, enter the following instructions.

```
> with(share): readshare(pvac,system):
```

These two commands cause Maple to load, from its Share Library, a file called *pvac* (print-vector-as-column), the effect of which is to change the way vectors are printed. If, for some reason, this functionality is to be switched off, enter the command

```
> pvac := false:
```

```
> print(V);
```

```
> pvac:=true:
```

```
> print(V);
```

In addition to noting how to turn this display feature on and off, observe that it takes **print** (or **evalm**) to get Maple to display the contents of a vector or matrix.

For the adventurous, from outside Maple, examine the file structure of the Maple V4 directory. There is a sub-folder called *Share* in which the contents of the Share Library are stored. A further sub-folder, *System*, contains another sub-folder called *Pvac*. In the *Pvac* folder there is a file Pvac.mpl, a file containing the code for the display feature being discussed. If this file can be rendered as a pure text file, it can be made into an initialization file so that the code will load automatically every time Maple is launched. The author of this manual has had this code running as an initialization file in both Release 3 and Release 4, a span of more two years. It has worked perfectly and has never given any trouble whatsoever.

To obtain a text version of the file pvac.mpl launch a text editor such as Word, etc. From within the text editor, locate the file pvac.mpl and open it. Perform a Save As, save the file as "text", give it the name Maple.ini (for the PC; on the Macintosh, use MapleInit, and for UNIX, use .mapleinit).

Be sure that your text editor does not add a hidden .txt or other such extension. Quit the editor, and drop the initialization file into the Lib sub-folder of the Maple V4 folder for the PC, drop it into the Maple V4 folder for the Mac, and experiment with where to put it on a UNIX system. Relaunch Maple and you will automatically have launched the *pvac* code.

To verify that a Maple vector behaves as a column vector, no matter what it looks like, try the following experiments.

```
> print(A,V);
> evalm(A &* V);
```

This is exactly the product you should obtain from the product $A \mathbf{V}$ done by hand, treating \mathbf{V} as a column vector. Now ask Maple to convert \mathbf{V} to a matrix.

```
> VC := convert(V,matrix);
> type(V,vector); type(VC,vector);
> type(V,matrix); type(VC,matrix);
> VC[1,1];
> VC[1,2];
> VC[2,1];
```

The matrix VC does not have a second column. It has two rows. If the vector \mathbf{V} is converted to a matrix data structure, it gets converted to a 2 x 1 (column) matrix. Maple thinks of the vector \mathbf{V} as a column thing, even if its default print style is to make it look like a row thing.

Moreover, the transpose of \mathbf{V} , if converted to a matrix, has all the characteristics of a 1 x 2 (row) matrix.

```
> VR := convert(transpose(V),matrix);
> VR[1,1];
> VR[1,2];
> VR[2,1];
```

The matrix VR does not have a second row. Maple thinks of the transpose of \mathbf{V} as a row thing, no matter how it prints it. In fact, it is a great tragedy that the Maple programmers have decided that the default output to the `transpose` command is

```
> transpose(V);
```

The evidence has already been presented that Maple understands the transpose. It is sad, indeed, that so fine a program as Maple should have such anomalous behavior when displaying the transpose of a vector \mathbf{V} that so obviously has the inherent properties of a column object.

Incidentally, this means that there is no way, per se, to enter a row vector into Maple. You enter a column vector, the default vector object, then transpose the column vector. And, yes, you live with not being able to see the display of the transpose as a row-like object.

Caution: It is tempting to sidestep this issue of row and column vectors with the belief that instead, row and column matrices will be used. This is not a good idea. There are commands in Maple that specifically demand vectors, not matrices. For example, if you had defined, not \mathbf{V} , but \mathbf{VC} , a column matrix, and wanted to live your Maple life with only matrices, you'd run afoul of

```
> dotprod(VC,VC);
```

So, you cannot live without vectors, and if you cannot live without vectors, you must then face the issue of row and column vectors. Sorry, but to reap the benefits of Maple you have to put up with a few quirks. Kind of like life in general.

There is one final issue to face about linear algebra in Maple. To perform operations on vectors one must map the operator onto the vector. For example, to simplify a vector, use the following syntax.

```
> V := vector([1/x+1/y, 1/x-1/y]);
> simplify(V);
```

Obviously, not the right syntax.

```
> map(simplify,V);
```

The operator **simplify** has to be mapped onto the vector.

As another example, if \mathbf{V} is a function of t and you want its derivative, you use the following syntax.

```
> V := vector([sin(t),cos(t)]);
> diff(V,t);
```

Obviously, the wrong syntax.

```
> map(diff,V,t);
```

Additional parameters to the mapped operator go at the end.

Unfortunately, there are two exceptions to the rule "Map things onto vectors and matrices."

```
> V := vector([Pi,2*Pi]);
> evalf(V);
> map(evalf,V);
```

Well, that seems to work. Why raise that as an exception? The command `evalf` can take an integer as a second argument, changing the number of digits returned.

```
> evalf(Pi,20);
```

But if you try that for the vector `V`, it fails.

```
> map(evalf,V,20);
```

Nonsense is returned. The method that works is

```
> evalf(op(V),20);
```

A second exception is substitution.

```
> V := vector([x,x^2]);
> subs(x=1,V);
> map(subs,V,x=1);
```

The syntax that works is

```
> subs(x=1,op(V));
```

Hence, the rule is "Map all operators except `subs` and `evalf`. For those, don't `map`, but use `op` around the vector. If you use `map`, don't use `op`. When `op` is needed, you don't use `map`."

1.2 On Line

The exercises of this section explore the concept of "span" by visualizing randomly generated members of the span of a set of vectors. Begin by loading both the `linalg` and `plots` packages.

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

Exercise 1

Enter into Maple the following four points. For most purposes, points can be represented as lists, a data structure denoted by square brackets.

```
> P1:=[1,1]; P2:=[1,-1]; P3:=[-1,1]; P4:=[-1,-1];
```

The *plots* package makes available a **pointplot** command that will plot a list of points. There are a number of options to this command that will vary the look of the graph, and by using the toolbars associated with the graph, many characteristics of the plot can be adjusted. To obtain help on this command, enter

```
> ?pointplot
```

To plot these points, enter

```
> pointplot([P.(1..4)]);
```

Exercise 2

Enter the vectors $\mathbf{A} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$.

```
> A:=vector([1,1]); B:=vector([2,3]);
```

Using the **pointplot** command, plot these two vectors as points in the plane. Note the additional parameter *view*, which sets a viewing window for the graph.

```
> pointplot([A,B],view=[0..3,0..3]);
```

a) Produce several different linear combinations of \mathbf{A} and \mathbf{B} . Appropriate syntax for doing this would be as follows.

```
> evalm(2*A + 3*B);
```

b) Use Maple's random number generator to create several random linear combinations of \mathbf{A} and \mathbf{B} . First, define a function f which generates random four-digit numbers in the interval $(-1,1)$. This is done with Maple's **rand** function as follows. The **evalf** command changes exact fractions to decimals.

```
> f:=evalf(rand(-10000..10000)/10000):
```

Then, invoke the function f with the syntax $f()$. For example, create \mathbf{c} , one random linear combination, with the syntax

```
> c:=evalm(f()*A+f()*B);
```

c) Plot enough points in the span of \mathbf{A} and \mathbf{B} to get a discernible geometric figure. Begin with 100 random linear combinations, and, using the **pointplot** command, plot them as points in the plane. Maple's **seq** command will produce a sequence of similar objects from a pattern provided to it. Terminate the command with a colon (:) to suppress the output. Then, feed the resulting sequence to the **pointplot** command, remembering to enclose the sequence in square brackets since **pointplot** requires a *list* of points (or vectors).

```
> q:=seq(evalm(f()*A+f()*B),k=1..100):
> pointplot([q]);
```

d) The plot in part (c) is only part of the span, the portion being determined by our use of random numbers in the interval $(-1,1)$. Rather than redefine the function f , try multiplying the vector \mathbf{A} by 2, creating another plot of at least 200 random linear combinations with \mathbf{A} multiplied by 2.

Exercise 3

Describe in words the set of points corresponding to the collection of linear combinations defined by the sum $s\mathbf{A} + t\mathbf{B}$, where both s and t lie in closed intervals of the form $[-2,2]$. Plot the resulting set of linear combinations as points in the plane, using a different color than used in Exercise 2. (See the online help for how to specify color in the **pointplot** command.)

Exercise 4

In Exercise 9 of the non-computer problems, it was stated that each element

$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ in the span of

$$\mathbf{X} = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \text{ and } \mathbf{Y} = \begin{bmatrix} -1 \\ 3 \\ 2 \end{bmatrix} \text{ satisfies } 5x + 3y - 2z = 0.$$

a) Write $\mathbf{C} = a\mathbf{X} + b\mathbf{Y}$, the expression for the general linear combination of \mathbf{X} and \mathbf{Y} . Then show that the components of the vector \mathbf{C} satisfy the equation of the plane declared above. This is most effectively accomplished as follows.

```
> X := vector([-1,1,-1]); Y := vector([-1,3,2]);
> C := evalm(a*X + b*Y);
```

```
> q := 5*x + 3*y - 2*z = 0;
> q1 := subs(x=C[1], y=C[2], z=C[3], q);
```

b) Plot, as points in R^3 , a few hundred elements of this span. This requires use of Maple's **pointplot3d** command, the 3d analog of **pointplot**. Begin by forming, via **seq**, a sequence of random vectors in the span of X and Y. The function f defined for Problem 1 can again be used to provide the random coefficients.

```
> q:=seq(evalm(f()*X+f()*Y),k=1..200):
```

The syntax for **pointplot3d** is similar to that of **pointplot**. However, there are several additional parameters whose use makes for a better graph. Since all 3d plot packages attempt to plot a 3d object on a 2d sheet of paper (or computer screen), there is a need for a reference frame that provides the sense of depth. Try putting a box around your graph, either interactively via the toolbars, or via options to the plot command itself. Having clear and highly visible labels on the axes is equally useful. Thus, the following syntax could be used to plot the vectors randomly generated above.

```
> pointplot3d([q], color=black, axes=boxed, labels=[x,y,z],
labelfont=[TIMES,BOLD,14]);
```

Observe that in Maple, 3d graphs can be rotated on the screen by manipulation with the mouse. Click on the plot to make it "live." Then, click and hold down the mouse button, dragging the bounding box that now replaces the graph. This bounding box is rotated as the mouse is moved. When released, the bounding box has a new orientation, and the graph is redrawn by clicking the **R** (redraw) on the toolbar.

Try to rotate your graph to demonstrate that the plotted points lie on a plane.

1.3 On Line

Maple Release 4 permits more than one worksheet to be open at the same time. On some platforms Maple can be put into the "multiple kernels" mode in which each worksheet is attached to its own "kernel," or memory state. In this mode, variables declared in one worksheet will not be known to any other worksheet opened simultaneously.

However, the default setting for Maple might be the "shared kernel" mode in which all open worksheets share the same memory state. In this case, variables declared in one worksheet have the same value in every other worksheet opened simultaneously. The potential here for grave confusion

is very high. Since, in Release 3, only one worksheet could be attached to a kernel, this conflict between multiple worksheets never arose. In Release 4 it is essential that this "gotcha" be understood. A simple precaution in the shared kernel world is starting every new worksheet with a **restart**, a Maple command which clears memory and resets all variables.

The use of the **restart** command at the beginning of each new worksheet is highly recommended.

Here, we both restart Maple and load the *linalg* package.

```
> restart;
> with(linalg):
```

Exercise 1

Obtain $\mathbf{X} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix} + s \begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} -3 \\ 0 \\ 1 \end{bmatrix}$ as the general solution to the equation $x + 2y + 3z = 4$.

Maple's **linsolve** command, the general linear system solver, will give the general solution to systems of equations. We begin by creating a matrix (the coefficient matrix) whose (i,j) entry is the coefficient of the j th variable in the i th equation. You can either type in the coefficient matrix A directly, or use the **genmatrix** command, giving a list of equations, and a list of variables as parameters. Thus, for the preceding linear equation, we could enter

```
> q := x+2*y+3*z = 4;
> A := genmatrix([q],[x,y,z]);
```

This extracts the coefficient matrix for the system. Next, the coefficient matrix and the constants on the right side of the equations are entered into **linsolve**. We express the constants as a list which, in this case, has only one entry.

```
> X := linsolve(A,[4]);
```

The arbitrary constants that Maple has introduced are $_t_1$ and $_t_2$. The lead character is the underscore (not a minus sign) and the numbers 1 and 2 are subscripts. To address these constants in Maple, use the syntax $_t[1]$ and $_t[2]$.

Setting these constants alternatively equal to 0 and 1 will "extract" the basis vectors in the general solution. After defining **u**, the translation vector, it is essential to subtract **u** from **X** when extracting the vectors **v** and **w**, the multipliers of $_t_1$ and $_t_2$.

```
> u := subs(_t[1]=0, _t[2]=0, op(X)); v := subs(_t[1]=1, _t[2]=0,
evalm(X-u)); w := subs(_t[1]=0, _t[2]=1, evalm(X-u));
```

To get Maple to write the general solution in the vector form given in the statement of the problem, adroit use of the **evalm** command is necessary. Unless **evalm** is applied to the vector **v**, the screen will merely display the name, **v**.

```
> Xg := evalm(u) + s*evalm(v) + t*evalm(w);
```

a) With the vectors **v** and **w** declared as above, (or typed in afresh from the pencil-and-paper solution), form **C**, the general element in the span of **v** and **w**.

```
> C := evalm(a*v + b*w);
```

b) Show that $\mathbf{F} = \mathbf{u} + \mathbf{C}$ solves the given equation $x + 2y + 3z = 4$.

```
> F := evalm(u+C);
```

```
> q;
```

```
> subs(x=F[1], y=F[2], z=F[3], q);
```

This shows that the general solution of the given equation is the span of the vectors **v** and **w**, translated by the vector **u**.

c) Substitute the vector **C** into the given equation and describe the result. What should you conjecture from this result? Can you prove it?

Exercise 2

Repeat Exercise 1 for the system (U) from Section 1.3. The essential question to be ultimately resolved is "Does the conjecture made in Exercise 1 still hold?" Can you then prove your answer is correct?

Begin by entering the system (U), being careful to clear the variable **w** that was used in Exercise 1,

using the command "**w:= 'w'**". Call your equations q1, q2, q3, and q4.

Use **genmatrix** to form **A**, the coefficient matrix for the system (U).

```
> A := genmatrix([q.(1..4)], [x,y,z,w]);
```

Use **linsolve** to obtain the general solution of system (U), using a list for the values on the right hand sides of the equations.

```
> X := linsolve(A, [1,0,1,2]);
```

Extract the translation vector **u** and the basis vectors **v** and **w** as in Exercise 1.

a) Form **C**, the general element in the span of **v** and **w**.

```
> C := evalm(a*v + b*w);
```

b) Show that $\mathbf{F} = \mathbf{u} + \mathbf{C}$ satisfies the system (U). This can be done by repetitive typing, by typing once and using copy/paste, or by a *for-loop* that does repetition automatically.

```
> F := evalm(u + C);
```

```
> for k from 1 to 4 do subs(x=F[1], y=F[2], z=F[3], w=F[4],
q.k); od;
```

c) Substitute C into each equation in the system (U) in an attempt to determine if the conjecture made in Exercise 1 is still viable. If you still believe your conjecture is true, can you prove it?

1.4 On Line

After restarting Maple and reinitializing by loading the *linalg* package, we examine the **rref** command for putting a matrix into its reduced (row) echelon form.

Consider the matrix A defined by

```
> A := matrix(4,6, [1,-1,1,3,0,6,2,-2,2,6,0,7,-1,1,1,-1,-2,1,4,-4,1,9,3,6]);
```

Subject A to the **rref** operator.

```
> rref(A);
```

The result just obtained is exact since Maple obtained it by doing rational arithmetic. There is no truncation error introduced by a conversion of integers to decimal form, and there is no round-off error produced by a numerical algorithm. This is the result that would be obtained working with a pencil and paper.

This is, of course, wonderful. We can, in principle, do every computation with total accuracy. At first glance, then, it seems then we should never again need to round off an answer. Unfortunately, life is not so simple. Suppose, for example, we want MAPLE to compute 1.048577^{20} .

We enter

```
> (1048577/1000000)^20;
```

Maple's response is a 121 digit integer divided by an equally large power of 10. Imagine now, what would happen if we attempted to perform a calculation which required, say, addition and multiplication several hundred such numbers. The number of digits our computer would need to store would become astronomical and the speed would be reduced to a snail's

pace. Furthermore, in an actual application, the number .1048577 would probably represent the result of some measurement which itself might be accurate only to within the given number of digits. Thus, the vast majority of the digits that our computer is so laboriously computing and saving are totally meaningless. The moral is that the perils of numerical computations must be faced.

Exercise 1

We first examine how to convert the matrix A to floating point (decimal) form, then look at the same row reduction done numerically instead of symbolically. The conversion can be done by *converting* each element to floating point form - **map** the convert operator onto the matrix A via

```
> A1 := map(convert,A,float);
```

Row reduce to reduced echelon form the floating point form of matrix A.

```
> rref(A1);
```

For the matrix A there is no difference in the reduced echelon form when working numerically. This will not always be the case. In fact, we can investigate Maple's numerics by a stratagem used on any numerical calculating device. Compute the value of $[\frac{1}{99}]99 - 1$, and successively append 9's both inside and outside the brackets. Eventually, there will be enough 9's so that the numeric calculation will no longer yield 0. That gives an indication of how accurate the computing device is. To force Maple to evaluate the expressions in floating point form, make one of the numbers a decimal. For example, use "1.0" rather than just "1" in the numerator of the fraction.

To see the difference between working numerically and symbolically in Maple, change the numerator from "1.0" to just 1. Then, Maple will evaluate the expression symbolically and produce 0. The round-off error only appears when working with floating point numbers.

It is possible to vary the number of digits with which Maple computes. This is done via the Digits variable as follows.

```
> Digits := 12;
```

Test Maple's numeric behavior on the floating point calculation above that failed to yield 0.

Notice that with more digits available, Maple escaped the effect of round-off in a computation that "failed" with just the default 10 digits.

The price one pays for increasing the number of working digits is computation time, since these extra digits are being simulated by the Maple software.

Reset the number of digits back to the default 10 via

```
> Digits := 10;
```

Exercise 2

Use the **rref** command to find all solutions to the system in Exercise 5g, Section 1.3.

First, enter the equations of that system. Call your equations q1, q2, q3, and q4.

Next, get Maple to write the augmented system matrix. The **genmatrix** command converts the equations into matrix form, and the additional parameter *flag* signals Maple to include the numbers on the right side of the equations. Incidentally, the parameter can be any character or word that is not already a reserved word in Maple.

```
> A:=genmatrix([q.(1..4)],[x,y,z,w],flag);
```

Apply the **rref** command.

```
> A1:=rref(A);
```

To obtain solutions from the rref form of the matrix A, apply the process of back substitution. Start with the bottom-most non-zero row of rref(A) and interpret it as an equation defining the value of *z*. Solve that equation for the value of *z* and substitute that value into the equation above. Solve the resulting equation for the value of *y* so determined. Substitute both the value of *y* and *z* into the remaining equation which is then solved for *x*.

Check your work by invoking Maple's built-in **backsub** command.

```
> backsub(A1);
```

Exercise 3

The rank of a system of equations is the number of equations left after eliminating dependent equations. This number does not depend on which equations were kept or eliminated. Hence, it is plausible that the rows in the reduced row echelon form of the system's matrix reflect the distinct equations that would survive an elimination of dependent equations. Hence, the rank of the system should be the number of non-zero rows in the reduced row echelon form of the matrix for the system.

Check this conjecture experimentally by creating (4 x 5) matrices A1, A2, A3, and A4 with ranks respectively 1, 2, 3, 4. In particular, insure that no matrix has a zero entry.

A process for creating a random matrix of prescribed rank is based on forming rows that are themselves linear combinations of other rows. Begin by defining `f`, a function returning a random integer in the closed interval [-10,10]. This is done with the **rand** command.

```
> f := rand(-10..10):
```

We begin by constructing a matrix A1, of rank 1. This requires that the rows of A1 be linear combinations of a single row. Begin by constructing a (1 x 5) matrix M1 by using the function `f` in conjunction with the **randmatrix** command to guarantee that the random matrix has entries that are integers in the interval [-10,10].

```
> M1 := randmatrix(1,5,entries=f);
```

From M1, build a (2 x 5) matrix M2 in which the rows are multiples of the single row in M1. Maple's **stack** command assembles a rows (or vectors) into a new matrix, making the building-blocks into the rows of the new matrix.

```
> M2 := stack(M1,evalm(f()*row(M1,1)));
```

From M2, build a (3 x 5) matrix M3 in which the rows are random linear combinations of the rows in M2. A single row in M2 can be referenced by the **row** command as illustrated below.

```
> M3 := stack(M2,evalm(f()*row(M2,1)+f()*row(M2,2)));
```

Finally, build the required (4 x 5) matrix A1 by taking linear combinations of the rows of M3.

```
> A1 := stack(M3,evalm(f()*row(M3,1) + f()*row(M3,2) + f()*row(M3,3)));
```

Test that A1 has rank 1 by invoking Maple's built-in **rank** command.

```
> rank(A1);
```

The process for constructing random matrices of rank 2 is similar. The only difference is that we start with a random (2 x 5) matrix (produced using the **randmatrix** command) instead of a (1 x 5) matrix. Similarly, for a rank 3 matrix, we would begin with a random (3 x 5) matrix and for a rank 4 matrix we would begin with a random (4 x 5) matrix.

The rank of the matrices A1, A2, A3 and A4 can be corroborated by reducing each to reduced echelon form. The following loop implements the required calculations.

```
> for k from 1 to 4 do rref(A.k); od;
```

In each case the number of distinct non-zero rows exactly matches the known rank of the matrix. These row reductions are exact, without round-off error since Maple computes symbolically unless told otherwise. However, all computing devices, when computing with floating point numbers, can experience difficulties attributable to round-off and truncation errors. Examine this issue in Maple.

```
> for k from 1 to 4 do rref(map(convert,A.k,float)); od;
```

For the matrices created in this session (remember, we are using a random process), `rref(A3)` is wrong. The `rref` command declares that a small number which ought to be seen as zero, is not zero. Hence, it suggests the rank of A3 is four.

One defense against such numeric errors is the `gausselim` command which row reduces a matrix but does not make the diagonal elements 1. By not dividing by the diagonal elements, this command is less likely to err in numeric computations. Let B3 be the floating point version of A3, obtained by mapping the process of conversion to floats onto A.

```
> B3 := map(convert,A3,float);
```

Now apply `gausselim`.

```
> q := gausselim(B3);
```

The small entries in the fourth row should be taken as 0's. These are the numbers that `rref` sees as non-zero, leading to errors. In Maple, we can apply the `fnormal` command to set to zero numbers smaller than a given tolerance. As with all operations applied to matrices and vectors, the `fnormal` command is mapped onto the matrix `q`.

```
> map(fnormal,q,10);
```

Exercise 4

Row reduce the transposes of the matrices A1, A2, A3, and A4 constructed in Exercise 3. The Maple command for the transpose is "transpose(A)"; What do you notice about the rank of the resulting matrices?

Exercise 5

Define vectors **X**, **Y**, and **Z** as indicated below.

```
> X := vector([1,2,-5,4,3]); Y := vector([6,1,-8,2,10]); Z
:= vector([-5,12,-19,24,1]);
```

a) Determine which of the vectors \mathbf{U} and \mathbf{V} below is in the span of \mathbf{X} , \mathbf{Y} , and \mathbf{Z} . Solve a determining system of equations by using the **rref** command.

```
> U := vector([-5,23,-41,46,8]); V := vector([22,0,-22,0,34]);
```

The question requires solving a $\mathbf{X} + b\mathbf{Y} + c\mathbf{Z} = \mathbf{U}$, and a $\mathbf{X} + b\mathbf{Y} + c\mathbf{Z} = \mathbf{V}$ for constants a , b , and c . Both sets of equations can be solved at the same time if the following augmented matrix is formed.

```
> q := augment(X,Y,Z,U,V);
```

Row reducing via the **rref** command gives solutions to both systems of equations at the same time.

b) Imagine that you are the head of an engineering group and that you have a computer technician working for you who knows absolutely nothing about linear algebra, other than how to enter matrices and commands into Maple. You need to tell your technician how to do problems similar to part (a) above. Specifically, you will give the technician an initial set of three vectors, \mathbf{X} , \mathbf{Y} , and \mathbf{Z} from R^3 . You will then provide an additional vector \mathbf{U} and you want the technician to determine whether \mathbf{U} is in the span of \mathbf{X} , \mathbf{Y} , and \mathbf{Z} .

Write a brief set of instructions which will tell your technician how to do this job. Be as explicit as possible. Remember that the technician cannot do linear algebra! You must provide instructions on how to construct the necessary matrices, what to do with them and how to interpret the answers. The final "output" to you should be a simple "Yes" or "No." You don't want to see matrices.

c) One of your assistant engineers comments that it would be easier for the technician in part (b) to use Maple's **rank** command rather than **rref**. What does your assistant have in mind?

1.5 On Line

After clearing Maple's memory by issuing a **restart** command, and re-initializing by loading the *linalg* package, enter the matrix \mathbf{A} and the vector \mathbf{X} .

```
> A := matrix(3,4,[1,2,1,3,-5,7,2,2,13,4,4,3]);
```

```
> X := vector([1,3,-2,4]);
```

Obtain the product $\mathbf{B} = \mathbf{A}\mathbf{X}$. (You should consult On Line Section 1.1 for a discussion of matrix products in Maple.)

```
> B:=evalm(A&*X);
```

Exercise 1

The matrix multiplication $A \mathbf{X}$ just obtained represents a linear combination of the columns of A , with coefficients taken from the vector \mathbf{X} . Implement this notion, and show the result is the vector \mathbf{B} found in the Introduction.

Columns of A can be referenced with the `col` command, and elements of the vector \mathbf{X} can be referenced as $\mathbf{X}[k]$. Hence, the brute force way of obtaining the required linear combination would be with the following syntax.

```
> evalm(col(A,1)*X[1] + col(A,2)*X[2] + col(A,3)*X[3] + col(A,4)*X[4]);
```

Since the columns of A are referenced in numerical order with an index that is repeated when referencing the components of \mathbf{X} , it should be possible to form the same linear combination of columns with some sort of summation process. Maple has a `sum` command that replicates exactly the mathematical sigma notation, $\sum_{k=1}^4 A_k X_k$. There is one syntactical quirk to overcome, however. The `col` command requires a value for the index before the `sum` command can provide it, so naive use of the notation will result in a syntax error. The trick is to put single forward quotes on the `col` command, thereby preventing it from demanding priority in getting a value of the index before the `sum` command is ready to provide it.

```
> evalm(sum('col(A,k)*X[k],k=1..4));
```

Exercise 2

Solve the system $A \mathbf{X} = \mathbf{B}$ for \mathbf{X} . Keep in mind that \mathbf{B} was formed by multiplying A against \mathbf{X} . This exercise seeks to determine whether or not \mathbf{X} can be recovered from \mathbf{B} .

One method of solution consists of row reducing the augmented matrix $[A, \mathbf{B}]$, then using back substitution, implemented in Maple via the `backsub` command.

```
> C := rref(augment(A,B));
> X1 := backsub(C);
```

By inspection, determine a value of the parameter $_t1$ that makes the general solution in $X1$ become precisely \mathbf{X} . Remember, this parameter is a subscripted quantity, and can be addressed in Maple via the syntax `_t[1]`.

Exercise 3

Another method for finding the general solution first obtained in Exercise 3 is predicated on finding a basis for the null space of A . This basis can be found via the Maple command `nullspace`, as shown below.

```
> q := nullspace(A);
```

Observe that the `nullspace` command returns a set of vectors. Here, there is but one member in the set, a single vector that can be addressed via the syntax

```
> Z:=q[1];
```

Verification that Z is indeed in the null space of A resides in the product $A Z$.

```
> evalm(A &* Z);
```

The general solution for the system $A X = B$ is then $Xg = X + t Z$, where t is an arbitrary parameter. Form Xg and show that it satisfies the equation $A Xg = B$.

By inspection, determine a value of the parameter t in Xg for which Xg becomes exactly $X1$, the first form of the general solution found above.

Exercise 4

Enter the matrix A and the vector B as shown below.

```
> A := matrix(4,6,[17,-6,13,27,64,19,4,-6,-33,25,7,9,55,-24,6,106,199,66,89,-36,32,160,327,104]);
```

```
> B := vector([17,4,55,89]);
```

a) Determine the rank of A . From this information, determine how many free variables the system $A X = 0$ will have.

b) How many spanning vectors will the null space of A contain?

c) Using Maple's `nullspace` command, find a spanning set for the null space of A . Since this command returns a set of vectors, extract all the vectors from this set, naming them $w1$, $w2$, etc.

```
> q := nullspace(A);
```

```
> for k from 1 to 4 do w.k := q[k]; od;
```

d) By inspection, find a vector X satisfying the equation $A X = B$. Verify that your guess indeed satisfies the equation.

e) If F is a general linear combination of the vectors $w1$, $w2$, ..., $w4$, show that $C = X + F$ is still a solution to the equation $A X = B$. (Note:

You might need Maple's **print** command as well as **evalm** to force Maple to display the results of your computations.)

f) Explain the statement "The general solution to $A \mathbf{X} = \mathbf{B}$ is the vector $\mathbf{X}_0 + W \mathbf{Y}$, where W is the matrix whose columns are \mathbf{w}_1 , \mathbf{w}_2 , \mathbf{w}_3 , and \mathbf{w}_4 , and \mathbf{Y} is any vector in R^4 ." Hint: Computationally, it would be useful to form the matrix W with the **augment** command, form the vector \mathbf{Y} with four parameters for components, and to find the general solution to $A \mathbf{X} = \mathbf{B}$ via the **linsolve** command. This solution should match $\mathbf{X}_0 + W \mathbf{Y}$.

g) Find a basis for the null space of A by solving the equation $A \mathbf{X} = \mathbf{0}$ for the general solution, \mathbf{X} . This is easily done in Maple by using the **linsolve** command. This command takes as arguments, the matrix A , and a vector (or list) of zeros as the right-hand side values. Maple will deliver a linear combination of the vectors \mathbf{w}_1 , \mathbf{w}_2 , ..., \mathbf{w}_4 that were found by the **nullspace** command.

h) Find a basis for the null space of A , this time solving the system $A \mathbf{X} = \mathbf{0}$ by augmenting A with a column of zeros and using the **rref** and **backsub** commands. The solution will not be readily recognized as a linear combination of the vectors \mathbf{w}_1 , \mathbf{w}_2 , ..., \mathbf{w}_4 .

i) Verify that the basis found in part (h) is equivalent to the basis $\{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_4\}$, show that the set of equations $a \mathbf{w}_1 + b \mathbf{w}_2 + c \mathbf{w}_3 + d \mathbf{w}_4 = v_k$ has a solution for each v_k in the basis found in part (h). This is most easily done by augmenting W with the general solution found in part (h), and using the **rref** command to show the equations are consistent for any values of the parameters in that general solution.

Chapter 2

Dimension

2.1 On Line

Restart Maple to clear its memory of all variables, and re-initialize by loading the *linalg* package.

Exercise 1

Given the matrix A entered below,

```
> A := matrix(4,3,[1,2,-3,4,5,-1,3,2,1,1,1,1]);
```

find the reduced (row) echelon form of A. How can you tell just from this reduced form that the columns of A are independent? Relate your answer to Theorem 1.

Exercise 2

Let A be a (random) matrix with more rows than columns. State a general rule for using $\text{rref}(A)$ to decide whether or not the columns of A are independent. Demonstrate your condition by (a) producing a 5 x 4 matrix A with no non-zero entries, and with independent columns; and by (b), producing a 5 x 4 matrix with no non-zero entries, and with dependent columns. In each case, obtain $\text{rref}(A)$. Prove your condition using Theorem 1.

Exercise 3

Let A be the matrix entered below.

```
> A := matrix(5,6,[-1,2,6,-8,-14,3,2,4,1,-8,5,-1,-3,1,4,-9,-10,0,3,-2,-1,12,-1,4,5,7,11,-11,-19,9]);
```

Use the `rref` command to find the pivot columns of A. Write them explicitly as columns. Then express the other columns of A as linear combinations of the pivot columns. (See Example 4 in the text.) You should discover that the first three columns of A are the pivot columns.

Exercise 4

If A_k represents the k th column of the matrix A defined in Exercise 3, form the matrix B whose columns are the columns of A in the following order: $B = [A_4, A_6, A_1, A_2, A_3, A_5]$. (This is most easily done by using the `augment` and `col` commands.) Find the pivot columns of B by using the `rref` command. Do you obtain a different set of pivot columns? Use `rref(B)` to express the other columns of B as linear combinations of the pivot columns. Could you have derived these expressions from those in Exercise 3? If so, how?

Exercise 5

Find a matrix C whose columns are just those of A listed in a different order, such that the column of C which equals A_5 and the column which equals A_1 are both pivot columns. Is it possible to find such a C where A_2 is a pivot column as well? If so, find an example. If not, explain why it is not possible.

2.2 On Line

Restart Maple to clear its memory of all variables, and re-initialize by loading the `linalg` package. In addition, use the command `"with(student):"` to load the `student` package in order to access its `equate` command.

Exercise 1

Let A be the matrix entered below.

```
> A := matrix(5,6,[-1,2,6,-8,-14,3,2,4,1,-8,5,-1,-3,1,4,-9,-10,0,3,-2,-1,12,-1,4,5,7,11,-11,-19,9]);
```

Part (a)

Find the rank of A via the **rank** command. Using only the value of the rank, explain why the statements *i*) and *ii*) given below are true.

i) The reduced form of the augmented matrix for the system $A \mathbf{X} = \mathbf{0}$ has three free variables. (Recall that in a previous On Line section it was noted that the rank is the number of non-zero rows in the reduced form.)

ii) The null space of A has dimension 3. (Hint: How many spanning vectors are there in the general solution to $A \mathbf{X} = \mathbf{0}$?)

Part (b)

Show that each of the vectors $\mathbf{X1}$, $\mathbf{X2}$, and $\mathbf{X3}$ given below satisfy $A \mathbf{X} = \mathbf{0}$.

```
> X1 := vector([-5,13,-10,2,-3,1]); X2 := vector([3,-6,11,1,2,-5]);
X3 := vector([-4,7,9,5,1,-6]);
```

Note: Since verifying that $A X_k = \mathbf{0}$ is a repetitive task, it can be done in a *for-loop*.

Part (c)

By computing the rank of the matrix $[\mathbf{X1}, \mathbf{X2}, \mathbf{X3}]$, prove that $\mathbf{X1}$, $\mathbf{X2}$, and $\mathbf{X3}$ are linearly independent. (Recall that the maximal number of linearly independent columns equals the rank.)

Part (d)

How does it follow that the dimension of the null space of A is 3? How does it follow that the

X_k constitute a basis for the null space?

Part (e)

Using Maple's **nullspace** command, find a basis for the null space of A . Express each vector in this basis as a linear combination of the X_k 's from part (d). Hint: Given two bases for this null space, showing that they are equivalent requires showing that any vector in one can be found as a linear combination of the vectors in the other. Hence, a set of equations of the form $a X_1 + b X_2 + c X_3 = w_k$ must be solved for each $k = 1, 2, 3$. This can be done by forming the augmented matrix $[\mathbf{X1}, \mathbf{X2}, \mathbf{X3}, \mathbf{w1}, \mathbf{w2}, \mathbf{w3}]$ and row reducing. In row reduced form this matrix will indicate whether or not these equations are solvable, and if so, how to express the non-pivot

vectors in terms of the pivot vectors. See Theorem 1 and Example 4 in Section 2.1.

Part (f)

In part (e), what made us so sure that the first three columns would be the pivot columns? Why couldn't, for example, the pivot columns be columns 1, 3, and 4? (Hint: Think about what this would imply for the reduced form of $[\mathbf{X1}, \mathbf{X2}, \mathbf{X3}]$.)

Part (g)

In part (e) you expressed the vectors w_k in terms of the vectors X_k . In this part, now express the X_k in terms of the w_k . This would complete the demonstration that the X_k and the w_k are equivalent spanning sets for the null space of A. Hint: Use the technique in part (e).

Part (h)

Find (by inspection) a vector \mathbf{T} which solves the equation $\mathbf{A} \mathbf{X} = \begin{bmatrix} 6 \\ 1 \\ 4 \\ 1 \\ 11 \end{bmatrix}$.

Part (i)

Let $\begin{bmatrix} r \\ s \\ t \end{bmatrix}$ be an arbitrary element of R^3 and let $\mathbf{Z} = \mathbf{T} + r \mathbf{X1} + s \mathbf{X2} + t \mathbf{X3}$, where \mathbf{T} is as found in part (h). Compute $\mathbf{A} \mathbf{Z}$. Explain why you get what you get. Find constants u , v , and w such that $\mathbf{Z} = \mathbf{T} + u \mathbf{w1} + v \mathbf{w2} + w \mathbf{w3}$. What theorem does this demonstrate?

Note: An efficient way of doing this is to use the **equate** command from Maple. One might first enter the two proposed expressions for \mathbf{Z} as follows:

```
> q1 := evalm(augment(X.(1..3))&*vector([r,s,t])); q2 := evalm(augment(w.(1..3))
```

We can equate these two vectors with the **equate** command from the *student* package and then solve for the required constants using the following syntax.

```
> q3 := equate(q1,q2);
> q4 := solve(q3,{u,v,w}); q5 := solve(q3,{r,s,t});
```

2.3 On Line

Restart Maple to clear all variables and reset its memory, then initialize by loading the *linalg* package.

Exercise 1

Using Maple's **randmatrix** command, construct M , a random 3×5 matrix. What do you expect for the rank of M ? Check, using the **rank** command. Is it conceivable that the rank could have turned out otherwise? Why is it unlikely? Finally, retain this matrix for use in the other exercises of this section.

Exercise 2

Form two different random linear combinations of the three rows of M , then append these two rows to M , thereby creating a 5×5 matrix. It helps to use **rand** to define a function f as a generator for the random coefficients needed for the linear combinations. Then, the **sum** and **row** commands simplify constructing the linear combinations of the rows of M . Finally, the **stack** command appends rows to the bottom of M . (See Exercise 3 from the On Line exercises for Section 1.4.) Using both **rank** and **rref**, test the rank of the enlarged matrix. What is the maximal number of linearly independent columns in the new matrix?

Exercise 3

For the 5×5 matrix created in Exercise 2, find a set of columns which forms a basis for the column space. Express the other columns as linear combinations of these columns. Use the technique of Example 3 in Section 2.1.

Exercise 4

In this exercise you will explore Maple's ability to obtain the reduced row echelon form numerically. In Exercise 3 the reduction is found symbolically, using exact arithmetic, and the result suffers no loss of accuracy from round-off error. A floating point evaluation of this exact answer will serve as the target answer that we will expect Maple to deliver numerically.

First, apply the **evalf** command to the reduced row echelon matrix found in Exercise 3. Next, by mapping the convert-to-float operation onto it, convert the 5×5 matrix of Exercise 2 to floating point form. (See

Exercise 1 in the On Line exercises for Section 1.4.) Then, obtain the reduced row echelon form of this numeric matrix. Observe that the result is wildly wrong. In fact, it shows the matrix to be of rank 4 whereas the matrix is known to have rank 3.

```
> mm := map(convert,MM,float);
> mm1:=rref(mm);
```

The reason for the error can be seen by row reducing the matrix to an upper-triangular form, without dividing the diagonal elements to make them 1's. This will prevent division by possible small numbers. This row reduction can be accomplished via the **gausselim** command.

```
> q := gausselim(mm);
```

So, although the work was done numerically, the only difficulty that has surfaced is the small positive value on the main diagonal, a value that in exact arithmetic would be zero. To get Maple to render such small values as zeros, map the **fnormal** command onto the matrix. The **fnormal** command takes as additional parameter, an integer specifying the number of digits to which the rounding is to be performed.

```
> rref(map(fnormal,q,9));
```

This result now matches what was obtained when the exact *solution* was converted to floating point form.

Exercise 5

Find a basis for the column space of the 5 x 5 of Exercise 2 by row reducing its transpose and invoking the Non-Zero Rows Theorem. Express each of the basis columns found in Exercise 3 as linear combinations of the basis columns found by this technique.

Exercise 6

Create four random 4 x 4 matrices of rank 1, 2, 3, and 4, respectively. (See Exercise 3 from the On Line exercises for Section 1.4.) Determine the null space and the dimension of the null space (called *nullity* in some texts) for each matrix. Relate the rank, the nullity, and the number of rows in the matrix.

Chapter 3

Transformations

3.1 Section 3.1 - On Line

Restart Maple to clear all variables, then load the *linalg* and *plots* packages. The exercises in this section deal with transformations in the plane. Hence, constructing several different types of graphs needed in this section requires the *plots* package.

Exercise 1

Create a $2 \times n$ matrix F whose n columns are the coordinates of certain points in the plane. These points are the endpoints of line segments that constitute a letter of the alphabet. Because it might be tedious to construct the letter "O" with line segments, feel free to select some more easily constructed letter, such as "F".

Sketch the letter of your choice on a piece of paper, using a minimum number of arcs, and a maximum number of line segments. Pick an endpoint of some segment as the origin and assign coordinates to each of the other endpoints. Sketching the letter on a sheet of graph paper might make this easier.

Enter into the matrix F the coordinates of the endpoints of the segments making up your letter. Each column represents an endpoint. Start at an extremity, and use contiguous columns to represent points connected by a line segment. If your letter requires you to retrace a segment (this happens with the arms in letters like E and F), list the coordinates of any endpoints in the order in which they are traversed.

For example, a recognizable letter F can be represented by the matrix

$$\begin{bmatrix} 0 & 0 & 2 & 0 & 0 & \frac{3}{2} \\ 0 & 2 & 2 & 2 & 1 & 1 \end{bmatrix}$$

where the origin has been taken at the base of the vertical stroke.

Save the worksheet in which you have entered your matrix F since it will be used in the remaining exercises.

Exercise 2

Since you will need (and want?) to plot your letter, create the following Maple function that takes as input the name of the matrix of your letter, and returns a plot of the letter. The effort required to type in this function will more than pay for itself as you experiment with these exercises.

```
> f := x -> pointplot([seq(convert(col(x,k),list),k=1..coldim(x))],
style=line, axes=boxed, scaling=constrained):
```

Obtain a plot of your letter by applying the function f to the variable F associated with the data for your letter.

```
> f(F);
```

Exercise 3

Example 1 from Section 3.1 contains a matrix M which represents a "shear along the x-axis." Call this shear matrix S_x and apply it to the letter stored in F by forming the matrix product

$S_x F$. Plot the image of the sheared letter.

Exercise 4

Construct the rotation matrix corresponding to a counterclockwise rotation of 20 degrees. Apply this rotation matrix to your letter and plot the result.

Since you will need other rotations, it will be more efficient if you build R, a "rotation matrix generating function" that accepts as input the number of degrees (counterclockwise) through which the rotation is to take place, and returns the matrix for this rotation. Maple's arrow notation for building functions is appropriate here. An appropriate syntax would be

```
> R := x -> matrix(2,2,[cos(x),-sin(x),sin(x),cos(x)]);
```

```
> R20 := R(20*Pi/180);
```

Rotate your letter by multiplying it by R_{20} and plot the rotated letter by invoking the plotting function f built in Exercise 2.

Exercise 5

Create another letter, reduce it to a matrix of coordinates, and store it in an appropriate variable. For example, the letter E can be created by a simple modification of the matrix representing the letter F, and its matrix stored in the variable E. Plot the new letter. Then, in anticipation of plotting the combination F E, determine a way to plot the result of shifting E three units to the right. Call your shifted letter TE.

Hint: Since the translate of E three units to the right would have each x-coordinate increased by 3, you want to find a simple way to add 3 to each element in the first row of the matrix E. The Maple syntax $3\$7$ writes a sequence of seven 3's separated by commas. Hence, the following matrix has row of threes and a row of zeros.

```
> T := matrix(2,7,[3$7,0$7]);
```

Next, plot the combination FE. The utility function f which we built for graphing letters is not sophisticated enough to accept multiple inputs the way the standard plot functions in Maple will. Hence, create plots of each letter and combine the resulting graphics objects with the **display** command. Assign the plot of each letter to a variable, being sure to terminate each command with a colon. Then invoke **display**.

```
> p1:=f(TE):
> p2:=f(F):
> display([p1,p2]);
```

Exercise 6

Let S be the transformation of R^2 to itself wherein $S(\mathbf{X})$ is a shift of \mathbf{X} one unit to the right. Show graphically that S is not linear. Specifically, use the letter created in Exercise 1 to show that $S(2 \mathbf{X}) \neq 2 S(\mathbf{X})$.

Exercise 7

For each of the following, find a matrix M for which multiplication would accomplish the indicated transformation. In each case, validate your matrix by applying it to the letter created in Exercise 1.

Part (a)

Ma flips a letter upside down.

Part (b)

Mb flips a letter left-to-right.

Part (c)

Mc rotates a letter by 20 degrees counterclockwise

Part (d)

Md is a shear along y-axis.

Exercise 8

Plot the effect of each of the following transformations applied to the letter created in Exercise 1.

Part (a)

A shear along the x-axis followed by rotation of 20 degrees counterclockwise.

Part (b)

Rotate 20 degrees counterclockwise, then shear along the x-axis.

Part (c)

Shear along the x-axis, followed by shear along the y-axis.

Part (d)

Shear along the y-axis, followed by a shear along the x-axis.

3.2 On Line

Restart Maple to clear its memory of all variables, then re-initialize by loading both the *linalg* and *plots* packages.

These exercises will continue the study the geometric aspect of transformations in R^3 . For this work it will be useful to again define the function *f* used in On Line Section 3.1. In that section the function took in a matrix representing a plane figure, and returned a plot of the figure.

Exercise 1

In place of the letters of the alphabet that were used in the exercises of Section 3.1, use line segments to create the outline of a car. Draw the outline on a piece of paper, even a sheet of graph paper, pick for the origin the endpoint of one segment, and find the coordinates of the endpoints of all the line segments. Enter these coordinates as columns of a matrix *C*. Hence, you might have a matrix such as the following.

```
> C := matrix([[0,1,3/2,5,6,0],[0,1,5/2,2,0,0]]);
```

representing the car whose shape is given by the following graph.

```
> f(C);
```

For the reader who believes this car looks more like a flat-iron, we give the data points for a car that is significantly better looking. The interested reader can enter the data into an appropriate matrix and run the experiments in these exercises with the improved image.

```
[[0, .954e-1], [.104e-1, .1612], [.1425, .2237], [.2124, .2336], [.2513, .2401],
[.2902, .2401], [.3187, .3158], [.3472, .3553], [.3912, .3684], [.4611, .3750],
[.5104, .3783], [.5674, .3783], [.6373, .3783], [.6710, .3684], [.6891, .3520],
[.7047, .3388], [.7202, .3191], [.7306, .2961], [.7409, .2763], [.7876, .2664],
[.8135, .2599], [.8264, .2500], [.8394, .2368], [.8472, .2072], [.8497, .1612],
[.8497, .1513], [.8497, .1283], [.8497, .1020], [.26e-2, .1053]]
```

Transform your plane image into a 3d object by altering the matrix *C* as follows. Add a middle row of zeros by first adding a third row of zeros, then swapping the second and third rows. This can be done in Maple by first stacking *C* with a row of zeros, then using the **swaprow** command to interchange the new row of zeros with the original row 2.

```
> d := swaprow(stack(C, [0$6]), 2, 3);
```

Since several 3d plots will be required, it is very useful to define a function *f3* that will take in a matrix representing a 3d object, and return a 3d plot of the object so represented.

```
> f3 := u-> pointplot3d([seq(convert(col(u,k),list),k=1..coldim(u))],
style=line, axes=boxed, scaling=constrained, color=black, labels=[x,z,y],
labelfont=[TIMES,BOLD,14]):
```

Having defined the function `f3`, apply it to the matrix `d` which represents a first version of a 3d car. Since this is a 3d plot, it can be rotated in Maple by clicking on the image and then using the mouse to "grab" and "rotate" the bounding box. Clicking on the `R` in the toolbar redraws the graph.

```
> f3(d);
```

Exercise 2

Further dimension can be added to the image of the car by adding $1/4$ to each element in row 3 of the matrix `d`, then augmenting the matrix `d` with the altered version `e`. One way to do this is to assemble (via `stack`) the first row of `d`, the altered second row, and then the third row.

```
> e := stack(row(d,1),evalm(row(d,2)+1/4),row(d,3));
> F := augment(d,e);
```

Test the efficacy of these improvements by plotting, using the function `f3`.

Exercise 3

Add some substance to the car by sketching in diagonal lines on each of the narrow faces. This requires alternating the columns of the augmented matrix `F` so that the first column comes from `d`, the second from matrix `e`, etc. In effect, build a new matrix `FF` by augmenting pairs of columns of the form $[d_k e_k]$. This is accomplished in Maple via the syntax

```
> FF := augment(F,seq(op([col(d,k),col(e,k)]),k=1..coldim(C))):
```

The validation of the manipulation is in the plotting.

```
> f3(FF);
```

Exercise 4

Some of the transformations that will be applied to the car include rotations. To keep the rotated car in the viewing window, it will help to move the origin to the center of the car. Deduce the coordinates of this center, and move the origin to that point by subtracting appropriate constants

from the first and third rows of the matrix `FF`. Remember that the first row records x-coordinates, the second row, z-coordinates, and the third row, y-coordinates. Form this new matrix by altering the appropriate rows and reassembling them into a new matrix.

```
> FFF := stack(evalm(row(FF,1)-3),row(FF,2),evalm(row(FF,3)-3/2)):
```

Next, rotate the figure counterclockwise 30 degrees about the x-axis, then rotate that image 20 degrees counterclockwise about the z-axis. This is most easily done by building functions that yield the appropriate three-dimensional rotation matrices (Exercises 10, 11, and 12 in Section 3.1), then invoking the functions for the required angles. (See Exercise 4 in the On Line section for Section 3.1.)

Exercise 5

Obtain a single matrix whose action under multiplication reproduces the two successive rotations implemented in Exercise 4. Validate your single matrix by again plotting the rotated car.

Exercise 6

What image would you see if you transformed the matrix `FFF` by a rank 2 transformation? Create a random rank 2 matrix and test your guess. After printing graphs of the transformed and untransformed images, attempt to label several points where the transformation is many-to-one. The entries in the transformation matrix should be random numbers in the range $[-1,1]$, lest the scale of the car be altered completely.

Exercise 7

What image would you see if you transformed the matrix `FFF` by a rank 1 transformation? Create a random rank 1 matrix and test your guess. Again, be sure to restrict the entries in your random matrix to the range $[-1,1]$.

3.3 On Line

Restart Maple to clear its memory of all variables, then reinitialize by loading the *linalg*, *plots*, and *plottools* packages.

Exercise 1

Create M , a random 2×3 matrix with rank 1. (See Exercise 3 in Section 1.4.) If M is interpreted as the matrix of a transformation acting on R^3 , what should the dimension of the image of this transformation be? Verify this by creating 100 random points in R^3 , transforming them under M , and plotting the points.

Next, generate P , a matrix containing 100 random points in R^3 . A moment's thought about how the transformed points will be plotted will determine the optimum strategy for generating and plotting the points. If the points are stored as columns of a matrix, they can be plotted by applying the **pointplot** command to the matrix. Hence, let P be of dimension 3×100 so the product $M P$ will be 2×100 . Create P by juxtaposing, via **augment**, 100 vectors generated by **randvector**. Terminating the commands with a colon (`:`) signals Maple not to print the rather large outputs to the screen. Finally, anticipating Exercise 2 where this plot will be required, store the plot data structure in a variable, say `p1`, so later, other images can be superimposed on it.

```
> P := augment(seq(randvector(3),k=1..100)):
> S := evalm(M&*P):
> p1 := pointplot(S):
> p1;
```

Exercise 2

The plot generated in Exercise 1 should show the span of any non-zero column of M . Demonstrate this by choosing a column of M and plotting, on the graph from Exercise 1, 100 random points in the span of this column.

Define the function `f` which generates a random integer in the closed interval $[-500,500]$. Then, using the **seq** command, give to **pointplot** a sequence of 100 random multiples of the first column of M . Color the points red and assign the plot data structure to a variable, say `p2`. After viewing the graph, merge it with the plot from Exercise 1 by use of the **display** command. Assign this composite graph to a variable, say `p3`, for use in Exercise 3.

```
> f := rand(-500..500):
> p2 := pointplot([seq(evalm(col(M,1)*f()),k=1..100)], color
= red): p2;
> p3 := display([p1,p2]): p3;
```

Exercise 3

Using information and insights from Exercises 1 and 2, find (a) a specific vector \mathbf{B} in R^2 for which the equation $M \mathbf{X} = \mathbf{B}$ is *not* solvable; and (b) a vector \mathbf{C} in R^2 for which the equation $M \mathbf{X} = \mathbf{C}$ is solvable. Indicate these vectors on the composite graph produced in Exercise 2. Verify your answers by computing, via `rref`, the reduced row echelon forms of the augmented matrices $[M, \mathbf{B}]$ and $[M, \mathbf{C}]$.

Exercise 4

Plot 100 random elements from the null space of M . Maple's `nullspace` command will provide a basis for the null space of M . This basis is returned as a set of vectors which can be extracted from the set with the bracket notation. The `seq` command can be used to generate a sequence of 100 random linear combinations of these basis vectors, a sequence which can then be plotted in R^3 with the `pointplot3d` command. The 3d graph so generated can be rotated on-screen by grabbing and rotating the bounding box. It should then be possible to observe the nature of that portion of the span so generated.

Finally, explain how this plot relates to the Rank-Nullity Theorem.

Exercise 5

Randomly generate a rank 2 matrix M of dimension 3×3 . Repeat Exercises 1 through 4, suitably modified to account for the different dimensions.

Specifically, this means you are to generate M . Then, in imitation of Exercise 1, the matrix P containing 100 random point in R^3 , and plot the product $M P$. In imitation of Exercise 2, plot 100 random linear combinations of the columns of M . In imitation of Exercise 3, find vectors \mathbf{B} and \mathbf{C} for which the systems $M \mathbf{X} = \mathbf{B}$, and $M \mathbf{X} = \mathbf{C}$ are not solvable, and solvable, respectively. Verify your choices of \mathbf{B} and \mathbf{C} computationally. Finally, in imitation of Exercise 4, plot 100 randomly chosen elements from the null space of M .

When you are done, don't forget to relate your findings to the Rank-Nullity Theorem.

3.4 On Line

Restart Maple to clear its memory of all variables, and re-initialize by loading the *linalg* and *student* packages.

Maple contains a variety of "solvers" for equations of various types. For example, the standard symbolic solver for one or several equations, linear and non-linear alike, is **solve**. The standard numeric solver for such equations would be **fsolve** (floating point solve). Differential equations are solved by **dsolve**, difference equations are solved by **rsolve** (recursive solve) and Diophantine equations are solve by **isolve** (integer solve).

In the *linalg* package, if a set of linear equations is captured in the matrix-vector format $A \mathbf{X} = \mathbf{B}$, then **linsolve** can be used. Both **solve** and **linsolve** return general symbolic solutions when applicable. In fact, **linsolve** will even return solutions in terms of arbitrary parameters. Other approaches to the solution of linear systems include the use of **gausselim** (followed by **backsub**) or **rref** (also followed by **backsub**). Maple's command structure is rich enough that nearly any undergraduate mathematics that can be articulated in standard mathematical notation can probably be implemented in the context of Maple's built-in commands.

There are two cautions to observe when using Maple to solve linear systems. First, if the calculation is done in floating point arithmetic, Maple is as liable to round-off and truncation errors as any other numeric utility. Second, when working symbolically, exact expressions for numbers can get dauntingly large, thereby consuming memory and time. Hence, Maple cannot solve symbolically systems as large as some strictly numeric utilities can solve by working in floats.

Exercise 1

Let A be the matrix from Exercise 2(a) of Section 3.4. Solve the system $A \mathbf{X} = \mathbf{B}$ where

$$\mathbf{B} = \begin{bmatrix} \frac{1}{10} \\ 21 \\ 32 \\ -44 \end{bmatrix}, \text{ then convert the answer to floating point form.}$$

Next, convert both A and \mathbf{B} to floats by mapping the `convert/float` operator onto them. Re-solve the system and compare the two floating point results. (See Exercise 4 in Section 2.3.)

Exercise 2

In many applications of linear algebra, numerical data comes from measurements which are susceptible to error. Suppose the vector \mathbf{B} in Exercise 1 was obtained by measuring a vector $\mathbf{B}\mathbf{a}$ whose actual value is $\mathbf{B}\mathbf{a} =$

$$\left[\frac{1}{100}\right] \begin{bmatrix} 210 \\ 321 \\ -440 \end{bmatrix}. \text{ Compute the solution to the equation } \mathbf{A}\mathbf{X}\mathbf{a} = \mathbf{B}\mathbf{a}.$$

Measure error is the absolute value of the difference between the computed value and the actual value. It can be computed in Maple with the following command.

```
> e := map(abs, evalm(X-Xa));
```

Which component of the solution \mathbf{X} computed in Exercise 1 has the largest error? (It might help to convert your answer to floating point form.) In terms of the magnitude of the components of the inverse $A^{(-1)}$, explain why this is the largest to be expected. (Maple computes the inverse of a matrix with the **inverse** command.) Which component of $\mathbf{B}\mathbf{a}$ would you change in order to produce the greatest change in $\mathbf{X}\mathbf{a}$? Why? Back up your answer with a numerical example or with a symbolic calculation where the increments in $\mathbf{B}\mathbf{a}$ are parameters successively appearing in each component. How much error could you tolerate in the measured values of the components of \mathbf{B} if the absolute value of the error in any entry of \mathbf{X} is to be at most .001?

Exercise 3

Let \mathbf{A} and \mathbf{B} be as defined below.

```
> A := matrix(3,3,[1,1/2,1/3,1/2,1/3,1/4,1/3,1/4,1/5]);
```

```
> B := vector([83,46,32]);
```

Find the solution to $\mathbf{A}\mathbf{X} = \mathbf{B}$. As in Exercise 2, suppose the vector \mathbf{B} was obtained by measuring a vector $\mathbf{B}\mathbf{a}$ whose actual value is $\mathbf{B}\mathbf{a} =$

$$\left[\frac{1}{100}\right] \begin{bmatrix} 8290 \\ 4607 \\ 3130 \end{bmatrix}. \text{ Solve the equation } \mathbf{A}\mathbf{X}\mathbf{a} = \mathbf{B}\mathbf{a}. \text{ What is the percentage}$$

error in the least accurate entry of \mathbf{X} ? How much error could you tolerate in the measured values of the components of \mathbf{B} if the absolute value of the error in any entry of \mathbf{X} is to be at most .001?

Exercise 4

Exercises 2 and 3 demonstrate that the process of solving a system of equations can "magnify" errors in disastrous ways. One quantitative measure of the inaccuracy of a calculation is the ratio of the percentage error in the final answer to the percentage error in the input data. But what do we mean by the percentage error in a vector (such as \mathbf{X} in Exercises 1 and 2) in which every component might have errors of different magnitudes?

For vectors in R^3 , this question has a geometric meaning. Think of X_1 and X_2 as representing points in R^3 . The distance d between these points is one measure of the error. If $X_1 = [x_1, y_1, z_1]^t$ and $X_2 = [x_2, y_2, z_2]^t$, then

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}.$$

In Maple, this can be computed as "norm($X_1 - X_2, 2$)". The additional "2" represents the "2-norm" wherein differences are squared and the square root of the sum taken. If X_1 is the computed answer and X_2 is the actual answer, we define the percentage error to be

$$P = 100 \text{ norm}(X_1 - X_2, 2) / \text{norm}(X_1, 2).$$

a) Let \mathbf{B} , \mathbf{Ba} , \mathbf{X} and \mathbf{Xa} be as in Exercise 2. Use the given formula to compute (i) the percentage error in \mathbf{B} , (ii) the percentage error in \mathbf{X} , and (iii) the ratio of the percentage error in \mathbf{X} to that in \mathbf{B} . This is the inaccuracy of the calculation of \mathbf{X} from \mathbf{B} . Assuming that accuracy is desired, do we want the number d to be large or small? Explain.

b) Compute the inaccuracy of the computation of \mathbf{X} from \mathbf{B} in Exercise 3.

c) For each $n \times n$ invertible matrix A there is a number "cond(A)" (the "condition number of A") such that the inaccuracy in solving the system $A \mathbf{X} = \mathbf{B}$ is at most cond(A), regardless of \mathbf{B} and regardless of the amount of error in \mathbf{B} . This means that if, say, cond(A) = 20 and the error in \mathbf{B} is .001%, then the computed value of \mathbf{X} will have at most $20 \times .001 = .02\%$ error. In general, $1 \leq \text{cond}(A)$. (This says that we cannot expect the answer to be more accurate than the input data.)

Maple has a built-in command for the condition number. Since the condition number is constructed from a "norm" of the matrix A , we need to specify the version of the condition number we want, according to the type of norm we want used. Hence, we will use the syntax cond(A,2). In addition, the 2-norm of a symbolic matrix can be a very large and complex expression. For this reason, we will only compute the condition number, based on the 2-norm, of matrices of floating point numbers.

Compute the condition numbers for the coefficient matrices in Exercise 1 and 3. Use this to explain the difference in accuracies obtained in these exercises.

Matrices with large condition numbers are called "ill-conditioned." If the coefficient matrix of a system is ill-conditioned, then we must be extremely suspicious of answers obtained by solving the system since any slight error in the input data can make the solution very inaccurate. Notice that these inaccuracies are not related to round-off error. Ill-conditioning is intrinsic in the matrix and not in the method of solution.

3.5 On Line

Restart Maple, clearing its memory of all defined variables. Then, re-initialize by loading the *linalg* and *student* packages.

These exercises will explore **LUdecomp**, Maple's built-in command for obtaining the LU decomposition of a matrix. The **LUdecomp** command can return seven different items, namely, L, U, a factorization of U into U1 and R, a permutation matrix P, the determinant of U1, and the rank of A. We will not need all of these outputs, and will restrict explorations to just L, U, and P.

The syntax for a multi-return Maple function is tedious. Each variable that is to have a return assigned to it must be included in the command surrounded by single quotes.

Since the actual return of the **LUdecomp** command is U, we will not need to make a specific request for U to be returned. Thus, to obtain L, U, and any permutation P needed to complete the decomposition, the appropriate syntax would be

```
u := LUdecomp(A, L = 'l', P = 'p');
```

Of course, if only L and U were desired, then the command could be shortened to

```
u := LUdecomp(A, L = 'l');
```

Exercise 1

Enter the Matrix A from Example 1 of the text, then obtain the LU decomposition by using the **LUdecomp** command. Assign the output of this command to the variable u, and let l be the lower triangular factor. This

matrix will not need a permutation P , so it can be omitted from the command. Finally, verify that $A = l u$. (You may need to use either **print** or **evalm** to view the contents of l .)

Exercise 2

Before examining the consequences pivoting has on the LU decomposition, we study the notion of a matrix factorization. When we demand that A be "factored" into the product LU , with L being a lower-triangular, and U being an upper-triangular matrix, we are asking for the solution of a set of equations. For example, if A is a given 3×3 matrix, then finding matrices L and U such that $A=LU$ is equivalent with solving a system of nine equations, where each equations is obtained by setting one entry of A equal to the corresponding entry of LU .

In this exercise, we investigate this system for the matrix A of Exercise 1. Begin by forming L and U , 3×3 matrices of indeterminates L_{ij} and U_{ij} .

Maple's **matrix** command, with appropriate if-statements as an option, will create the desired matrices.

```
> L := matrix(3,3,(i,j)-> if i<j then 0 else L.i.j; fi); U
:=matrix(3,3,(i,j)-> if i>j then 0 else U.i.j;fi);
```

Multiply L and U , forming a template of indeterminates which we then demand reduce to the entries of A from Exercise 1, above. This gives a set of nine equations in twelve unknowns, since there are six unknowns in each of L and U . The solution of this set of equations will not be unique, suggesting that we can impose an additional three conditions on the factorization.

```
> LU := evalm(L &* U);
```

The product LU and the matrix A can be equated via the **equate** command from the *student* package. This command returns a set of equations formed by equating corresponding entries of each matrix.

```
> q := equate(LU,A);
```

Typically, the **solve** command needs a set of equations and a set of variables. If no variables are suggested to Maple, it will attempt to deduce what the unknowns of the problem actually are. That is convenient here since it would be tedious to enter the names of all the variables in these equations.

```
> q1 := solve(q);
```

It is clear that we did not get a unique solution for the entries of L and U . Careful inspection shows there are three indeterminates in the answer.

This becomes more evident if we substitute these solutions into the matrices L and U.

```
> L1 := subs(q1,op(L)); U1 := subs(q1,op(U));
```

Exercise 3

Change the definition of the matrix L used in Exercise 2. Since there are three free parameters in the solution for the factors L and U, choose to have the diagonal elements of L all be 1's. This can be done with an appropriate if-statement in the **matrix** command that defines L. The matrix U will be the same as used in Exercise 2. Repeat the formation of nine equations in nine unknowns, obtaining unique factors L and U for the matrix A in Exercise 1. Display the resulting matrices L and U, and show that they are exactly the factors produced by the **LUdecomp** command in Exercise 1.

Exercise 4

In this exercise you will explore the concept of a permutation matrix P whose rows are a permutation of the rows of the identity matrix. If a matrix A is multiplied by P, the rows of PA will be permuted in the same way that the rows the identity were when forming P. Let P be a permutation of the rows of the 4x4 identity. This can be done in Maple by stacking a sequence of rows from the identity. Create A, a random 4x4 matrix and examine A, PA, and P.

```
> Id := diag(1$4);
> P := stack(seq(row(Id,k), k = [2,4,3,1]));
> A := randmatrix(4,4);
> PA := evalm(P &* A);
> print(A,PA,P);
```

Finally, observe that for a permutation matrix P, the inverse is the transpose.

```
> print(inverse(P),transpose(P));
```

Exercise 5

We wish to study the effect of pivoting on the LU decomposition. For this, we need a matrix A that forces a pivot. If the (1,1)-element of A were to be zero, a pivot would have to be performed immediately, since the row-reduction process by which we obtain L and U is basically gaussian elimination. Create A , a random 4x4 matrix, and then re-assign its (1,1)-element the value zero.

Use the **LUdecomp** command to obtain the LU decomposition, this time including parameter $P = 'p'$ so that a permutation matrix p will be returned. To have Maple display p , l , and u side-by-side, use the **print** command.

```
> u := LUdecomp(A,L='l',P='p'): print(p,l,u);
```

Use the **print** command to display, side-by-side, the product lu , the matrix A , and the product plu .

You should observe that the product lu does not reproduce A . That is the effect of pivoting. Whenever rows must be interchanged during the factorization, these interchanges are recorded in the matrix P . The resulting LU factorization is then a factorization of $P^{(-1)} A$, not A , resulting in the equality $P^{(-1)} A = LU$. Thus, $A = PLU$, not just LU .

Chapter 4

Orthogonality

4.1 On Line

Restart Maple to clear its memory of all defined variables, and re-initialize by loading the *linalg*, *plots*, *student*, and *plottools* packages.

Given the vectors $Q_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $Q_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, we will generate a coordinate grid corresponding to a space in which Q_1 and Q_2 are the basis vectors. We will plot the skewed grid lines in red, atop a standard coordinate system in black.

```
> Q1 := vector([-1,1]); Q2 := vector([1,2]);
```

The vector equation of a line through the tip of the vector Q_1 and parallel to the vector Q_2 is given by $\mathbf{R}(t) = Q_1 + t Q_2$. To form parallel lines through the tip of $2 Q_1$, $3 Q_1$, etc., we need to form the vectors $k Q_1 + t Q_2$, with k an integer in some interval $[-a,a]$. Alternatively, the equations of lines parallel Q_1 through the tip of Q_2 , $2 Q_2$, etc, we form the vectors $k Q_2 + t Q_1$. In every case we let t , the parameter of the line, range over an interval $[-b,b]$.

Maple's `seq` command can be used to generate an appropriate sequence of vector representations of the grid lines. Plotting them in red will produce the desired grid.

A template for the vector form of each set of grid lines is obtained as

```
> P1 := evalm(k*Q1+t*Q2); P2 := evalm(k*Q2+t*Q1);
```

Sequences of equations of the skew grid lines are formed with the `seq` command.

```
> s1 := seq([P1[1],P1[2],t=-5..5],k=-5..5): s2 := seq([P2[1],P2[2],t=-5..5],k=-5..5):
```

The plot of the grid lines is assigned to the variable `f1` so that it can be used again in one more activity. We include the scaling parameter in the `plot` command. A 1-1 scaling can also be imposed interactively from the toolbar. A view window is also set in the `plot` command.

```
> f1 := plot({s1,s2}, color=red, scaling=constrained, view=[-6..6,-6..6]):
f1;
```

For a finishing touch, we use the `arrow` command from the `plottools` package to draw Q_1 and Q_2 , the basis vectors of the skew coordinate system. We plot the first in green and the second in blue, superimposing both on the skewed red grid.

```
> a1 := arrow([0,0],[1,2], .2, .4, .2, color=green): a2 := arrow([0,0],[-1,1], .2, .4, .2, color=blue):
display([f1,a1,a2]);
```

Exercise 1

Modify the commands in the Introduction to produce red grid lines corresponding to the basis vectors $Q_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and $Q_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$. In addition, produce a graph showing the skewed grid lines and the basis vectors in green and blue. Assign this graph to a variable so that it can be re-used in Exercise 4.

Exercise 2

The curve defined implicitly by the equation $x^2 - \frac{y^2}{4} = 1$ is a hyperbola that will be plotted in Exercise 3. Here, show that in the basis of Exercise 1 (with coordinates u and v) the equation of this hyperbola is $4uv = 1$.

Begin by deducing the equations of the transformation. The point matrix M is constructed with columns Q_1 and Q_2 . Letting $\mathbf{X} = \begin{bmatrix} x \\ y \end{bmatrix}$ and $\mathbf{U} = \begin{bmatrix} u \\ v \end{bmatrix}$, the transformation equations are $\mathbf{X} = M \mathbf{U}$.

The vectors \mathbf{X} and $M \mathbf{U}$ can be equated with the `equate` command from the student package. The equation $\mathbf{X} = M \mathbf{U}$ defines the change of basis from xy -coordinates to uv -coordinates. Once these equations are obtained, Maple's `subs` command can be used to impose this change of coordinates on the equation of the hyperbola. An appropriate syntax might be

```
> q1 := equate(X, MU);
```

```

> q2 := x^2 - y^2/4 = 1;
> q3 := subs(q1,q2);
> q4 := simplify(q3);

```

Exercise 3

Use Maple's **implicitplot** command from the *plots* package to obtain a graph, in the xy -coordinates system, of the original hyperbola. Assign the graph to a variable so that in Exercise 4 it can be re-used. Be sure to use 1-1 scaling so that no distortion is introduced by the computer screen.

```

> g3 := implicitplot(q2,x=-10..10,y=-10..10,color=black, scaling=constrained):
g3;

```

Exercise 4

Use Maple's **display** command to superimpose on the skewed grid lines of Exercise 1, the graph of the hyperbola from Exercise 3. Then use Maple's digitizer (click on the graph, click on a point in the graph, read the coordinates in the window at the top-left of the graphics toolbar) to approximate the coordinates of some point on the hyperbola. By counting and estimating, infer the corresponding uv -coordinates. Show that to within the accuracy of the digitizer, your coordinates satisfy the equation of the hyperbola in the xy -system and in the uv -system.

For example, the point (2,4) appears to be almost on the hyperbola. The digitizer gives (2, 3.5). The same point appears to be (2, 0.1) in the skew grid. Hence,

```

> subs(x=2,y=3.5,q2);
> subs(u=2,v=.1,q4);

```

4.2 On Line

Restart Maple, clearing its memory of all defined variables. Then, re-initialize by loading the *linalg* and *student* packages.

You are working for an engineering firm and your boss insists that you find one single solution to the following system:

$$2x + 3y + 4z + 3w = 12.9$$

$$4x + 7y - 6z - 8w = -7.1$$

$$6x + 10y - 2z - 5w = 5.9$$

You object, noting that

(a) The system is clearly inconsistent: the sum of the first two equations contradicts the third.

(b) You need at least four equations to determine four unknowns uniquely. Even if the system were solvable, you couldn't produce just one solution.

The boss won't take "no" for an answer. Concerning (a), the boss points out that the system was obtained from measured data and any inconsistencies can only be due to experimental error. Indeed, if any one of the constants on the right sides of the equations were modified by .1 units in the appropriate direction, the system would be consistent.

Concerning (b), the boss says "Do the best you can. We will pass this data on to our customers and they wouldn't know what to do with multiple answers."

After some thought, you realize that projections can help with the inconsistency problem. The given system can be written in vector format as

$$x \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} + y \begin{bmatrix} 3 \\ 7 \\ 10 \end{bmatrix} + z \begin{bmatrix} 4 \\ -6 \\ -2 \end{bmatrix} + w \begin{bmatrix} 3 \\ -8 \\ -5 \end{bmatrix} = \begin{bmatrix} 12.9 \\ -7.1 \\ 5.9 \end{bmatrix}$$

You realize that this system would be solvable if the vector on the right side of the above equation were in the space spanned by the four vectors on the left. Using Maple's rank command, you quickly compute as 2, the rank of the system matrix A,

$$A = \begin{bmatrix} 2 & 3 & 4 & 3 \\ 4 & 7 & -6 & -8 \\ 6 & 10 & -2 & -5 \end{bmatrix}$$

showing that these four vectors in fact span a plane. Call this plane W.

Your idea is to let \mathbf{Bw} be the projection of $\mathbf{B} = \begin{bmatrix} 12.9 \\ -7.1 \\ 5.9 \end{bmatrix}$ onto W.

Since the system is so nearly consistent, \mathbf{Bw} should be very close to \mathbf{B} . Furthermore, the system $A\mathbf{X} = \mathbf{Bw}$ should certainly be solvable and one of the solutions should be what the boss is looking for.

Your point (b) will require some further thought. However, you do eventually come up with an idea which will be described in the exercises which follow.

Before going on to the exercises, it will be useful to enter the data of this system of equations. Enter the matrix \mathbf{A} and the vector \mathbf{B} . Convert \mathbf{B} to exact rational form and call that vector \mathbf{b} . Compute the rank of \mathbf{A} , verifying that it is indeed 2.

```
> A := matrix(3,4,[2,3,4,3,4,7,-6,-8,6,10,-2,-5]);
> B := vector([12.9,-7.1,5.9]);
> b := map(convert,B,rational);
> rank(A);
```

Exercise 1

Find an orthogonal basis for the column space of \mathbf{A} , use the Fourier Theorem to obtain \mathbf{Bw} , the projection of \mathbf{B} onto W , the column space of \mathbf{A} . Then solve $\mathbf{A}\mathbf{X} = \mathbf{Bw}$, expressing the solution in parametric form. Write \mathbf{X} as a sum of a "translation" vector, and vectors in the null space of \mathbf{A} . Show that the translation vector Maple finds is not orthogonal to the null space of \mathbf{A} .

First, obtain an orthonormal basis for the span of the column space of \mathbf{A} . Since the columns of \mathbf{A} are not linearly independent (\mathbf{A} has rank 2), determine (via `rref`) which two columns of \mathbf{A} to take as independent, and pass those two vectors to Maple's `GramSchmidt` command for orthogonalization. This command returns a list of *lists* (not *vectors*), so map the `convert-to-vector` operator onto this output. You will now have a list of two orthogonal vectors. An appropriate syntax might be:

```
> q := map(convert,GramSchmidt([col(A,1),col(A,2)]), vector);
```

Next, apply the Fourier Theorem to get \mathbf{Bw} , the projection of \mathbf{B} onto W , the column space of \mathbf{A} . Work with \mathbf{b} , the exact version of the vector \mathbf{B} . The formula in the Fourier Theorem is implemented in Maple much as it is written mathematically. Reference the basis vectors as `q[1]` and `q[2]`, compute dot products with Maple's `dotprod` command, and norms with Maple's `norm` command, being sure to compute the 2-norm.

Next, solve the system $\mathbf{A}\mathbf{X} = \mathbf{Bw}$. Maple's `linsolve` command will yield the general solution.

This general solution can be put into the form $\mathbf{X} = \mathbf{T} + \alpha \mathbf{v1} + \beta \mathbf{v2}$, where the vectors $\mathbf{v1}$ and $\mathbf{v2}$ are in the null space of \mathbf{A} . These vectors can be extracted from this general solution by an adroit use of substitution via the `subs` command.

Exercise 2

Objection (b) amounts to this: If the system is inconsistent, there is no solution. If the system is consistent, there are many solutions. Even the use of projections in Exercise 1 has yielded a general solution that is not unique. Perhaps your first thought was to report the translation vector as the solution. But there is nothing special about this vector, and the Translation Theorem says the general solution can be expressed using any particular solution, not just the translation vector.

Your next idea, however, is sound. With \mathbf{T} as the translation vector, let \mathbf{Tn} be its projection onto the null space of A . Report to your boss the solution $\mathbf{X} = \mathbf{T} - \mathbf{Tn}$. Why is this \mathbf{X} a solution?

Use Maple's **nullspace** command to find a basis for the null space of A . Then use the Fourier Theorem to obtain the projection of \mathbf{T} onto this null space. Finally, form $\mathbf{T} - \mathbf{Tn}$ and explain why this is a solution. Note, however that the Fourier Theorem assumes an orthogonal basis for the space into which the projection occurs. Hence, you will need to apply Maple's **GramSchmidt** command to the basis for the nullspace. Remember, though, that **GramSchmidt** returns a list of lists, requiring us to convert the sub-lists back to vectors as was done in Exercise 1.

Exercise 3

Try computing \mathbf{X} in Exercise 2 by starting with a solution other than \mathbf{T} . You should get the same \mathbf{X} . Why? It can be shown that the \mathbf{X} found in Exercise 2 is the solution of minimal length.

Exercise 4

Show that of all solutions to $A\mathbf{X} = \mathbf{Bw}$, the one with minimal length is the solution \mathbf{X} computed in Exercise 3. For this, let $\mathbf{q1}$ be the general solution found by Maple in Exercise 1, considered as a two-parameter family of vectors. Compute the 2-norm, and use calculus to minimize this norm. The resulting solution should be the same \mathbf{X} that was computed in Exercise 3.

Hint: Use the **subs** command to replace the free parameters $_t[1]$ and $_t[2]$ used by Maple with a and b , calling the result \mathbf{Xg} . Now, obtain the 2-norm of the vector \mathbf{Xg} . Since \mathbf{Xg} is a symbolic vector, Maple returns the norm with absolute values, thereby making it difficult to differentiate and set derivatives equal to zero. Simplify the norm of \mathbf{Xg} , adding the parameter *symbolic* to coax Maple to simplify the absolute values.

```
> f := simplify(norm(Xg,2),symbolic);
```

Differentiate with respect to a and b . Use Maple's **solve** command to solve the system obtained by setting these partials equal to zero, as follows. Finally, substitute these values into \mathbf{Xg} and compare with \mathbf{X} .

```
> fa := diff(f,a); fb := diff(f,b);
> qq := solve({fa,fb},{a,b});
```

Exercise 5

Maple would have found the least squares solution of minimal norm with its built-in **leastsqrs** command. Verify that this command yields the solution \mathbf{X} found in Exercise 2. Note the inclusion of the parameter *optimize* which signals Maple to find the solution of minimal length. Without this parameter, the **leastsqrs** command will return the general solution found in Exercise 1 as the vector in `q1`.

```
> leastsqrs(A,b,optimize);
> leastsqrs(A,b);
```

Exercise 6

After giving the boss your answer, you delete all your data except for \mathbf{A} and \mathbf{X} . A month later the customer calls, saying, "We know that there must be other solutions. Could you please provide us with the general solution?"

Show how the general solution can be reconstructed from \mathbf{A} and \mathbf{X} with a single Maple command.

4.3 On Line

Restart Maple to clear its memory of all previously defined variables. Then, re-initialize by loading the *linalg* and *plots* packages. In addition, enter the following lines of Maple code. This code, written by Dr. Mike Monagan of Simon Fraser University in Burnaby, British Columbia, Canada, creates a function that will generate the periodic extension of a function. The code first appeared in the article Tips for Maple Users and Programmers, *MapleTech*, VOL. 3, NO. 3, 1996, published by *Birkhauser*.

```
PE := proc(f, d::range) subs({'F' = f, 'L' = lhs(d), 'D' = rhs(d)-lhs(d)},
proc(x::algebraic) local y; y := floor((x-L)/D); F(x-y*D); end) end;
```

These lines of code can be entered into a separate Maple worksheet, and that worksheet saved. Later, if the code is again needed, that worksheet can be opened, and the lines copied and pasted into the active worksheet. There are, of course, other ways of saving code and making it accessible more easily, but some aspects of that process are platform dependent and will not be discussed here.

Exercise 1

Figure 1 of Section 6.6 in the text depicts a saw-tooth function called a "rasp." Plot the first, fourth, and tenth Fourier sine approximations to this function.

The rasp is generated by the periodic extension of the function $f(x) = x$, for x in the interval $[-1,1]$. To get Maple to plot the periodic extension of $f(x)$, use the function **PE** defined in the Introduction above. First, be sure to define $f(x)$ with Maple's arrow notation, thereby making f a function, not an expression.

```
> f := x -> x;
```

Define the function whose name is "rasp" as the periodic extension (hence, **PE**) of the function whose name is f . Do this by applying the **PE** operator to f , being sure to terminate the command with a colon ($:$) since the output will look strange, and probably unintelligible. The arguments to **PE** are the function to be extended, and the domain of the function being extended.

```
> rasp := PE(f,-1..1):
```

Obtain a graph of the rasp on the interval $[-3,3]$, assigning the plot to a variable for use later. The plot option $discont = true$ signals Maple to observe the discontinuities in the function, and tells it not to connect across the jumps.

```
> f1:=plot(rasp(x),x=-3..3, discont=true, color=black, scaling=constrained,
thickness=3): f1;
```

Obtain the Fourier sine series coefficients $b_n = \left[\frac{2}{l}\right] \int_0^l f(x) \sin\left(\frac{n\pi x}{l}\right) dx$. Here, $l = 1$ and $f(x) = x$. An integral can be entered into Maple with the **Int** command which stores the integral as an unevaluated symbol. If instead, the integral is entered with the **int** command, the evaluation of the integral is immediate. Here, **Int** is used so that the integral will be displayed completely.

```
> q := 2*Int(x*sin(n*Pi*x),x=0..1);
```

To evaluate an integral that has been entered with **Int**, apply Maple's **value** command to the integral.

```
> q1 := value(q);
```

We wish to simplify this expression. For example, $\sin(n\pi)$ is zero whenever n is an integer. We will tell Maple that n is an integer with its **assume** command. However, that will cause Maple to attach a tilde (\sim) to each n it prints thereafter. Suppressing the attachment of the tildes can be done either interactively from the Options menu (Options, Assumed Variables, No Annotation) or from the command line with the following **interface** command.

```
> interface(showassumed=0);
```

Now, use the **assume** command to tell Maple that n is an integer.

```
> assume(n,integer);
```

If the Fourier coefficients are now simplified, they will appear much like they would if the calculation were done "by hand." Note how, in the interest of simplicity, we assign the result to the name "b" and not to something that tries to reflect the dependence on n .

```
> b := simplify(q1);
```

The Fourier approximations are simply partial sums of the Fourier series. The first approximation, $p1$, is just $b_1 \sin(\pi x)$ and the tenth one is $p10 = \sum_{n=1}^{10} b_n \sin(n\pi x)$. We can obtain these expressions in Maple by using its **sum** command.

```
> p1:=sum(b*sin(n*Pi*x),n=1..1); p4:=sum(b*sin(n*Pi*x),n=1..4);
p10:=sum(b*sin(n*Pi*x),n=1..10);
```

Finally, these three partial sums can be plotted with a single **plot** command by grouping the functions in a list. Colors can be assigned to the functions by the option *color* = [...], with matching colors being listed in the order of the functions to which they are being ascribed. Assigning the plot to a variable allows merging, via the **display** command of the *plots* package, the approximations with the graph of the rasp created above

```
> f2 := plot([p1,p4,p10],x=-3..3,color=[red, green, blue]);
display([f1,f2]);
```

Exercise 2

Let $g(x) = \begin{cases} -1 & x < 0 \\ 1 & 0 \leq x \end{cases}$, a piecewise defined function. Obtain Fourier sine approximations with 4, 8, and 20 sine functions.

The periodic extension of $g(x)$ is called a "square wave." Note the "ear-like" peaks which appear in the graph of the partial sums at the discontinuities of $f(x)$. These peaks are referred to as the "Gibbs phenomenon." They are quite pronounced, even after twenty terms of the Fourier series. Their existence shows that it takes a very high fidelity amplifier to reproduce a square wave accurately. For this reason, square waves are sometimes used to test the fidelity of an amplifier.

Begin by defining $g(x)$ as a piecewise function on the interval $[-1,1]$. Use Maple's **piecewise** function which permits the definition a function with multiple formulas. Define g as a function by using the arrow notation.

```
> g := x -> piecewise(x<0,-1,x>=0,1);
```

Check the behavior of $g(x)$ by plotting it, again using the plot option `discont = true` so that jumps in the function are not connected.

```
> plot(g(x),x=-1..1,discont=true, color = black);
```

Define G as the periodic extension of the function g . Use the PE code detailed in the Introduction. Again, end the command with a `clon (:)` since the echo will probably not make much sense to the typical student of linear algebra. Plot G on the interval $[-3,3]$, assigning the plot to a variable for use later.

As in Exercise 1, the Fourier sine coefficients b_n are computed by integrating $g(x)$. Enter the defining integral using **int** and $g(x)$. The presumption here is that the **interface** and **assume** commands are still operative from Exercise 1. If not, re-execute those commands.

Exercise 3

Obtain the thirtieth partial sum of the Fourier sine series for the function $f(x) = \frac{1}{1+x^2}$, $-1 \leq x \leq 1$, then graph the periodic extension of $f(x)$ and the Fourier approximation. Explain why the graphs don't agree. (See Exercises 6 and 7 of Section 6.6).

Exercise 4

Obtain a Fourier cosine series for the function in Exercise 3. Plot the first, fourth, and eighth partial sums.

4.4 On Line

Restart Maple to clear its memory of all defined variables, and re-initialize by loading the *linalg* and *plots* packages.

Let $R_x(\frac{\pi}{6})$ be the matrix of a counterclockwise rotation, around the x -axis and through an angle of $\frac{\pi}{6}$ radians. Let $R_y(\frac{\pi}{4})$ be the matrix of a counterclockwise rotation, around the y -axis and through an angle of $\frac{\pi}{4}$ radians. Let $A = R_x(\frac{\pi}{6})R_y(\frac{\pi}{4})$. Since the product of two orthogonal matrices is orthogonal, A is orthogonal. The purpose of these exercises is a demonstration that A defines a rotation about a fixed axis and through a particular angle. See Figure 5 in Section 4.4 of the text..

Points on this axis remain fixed under the rotation. Thus, if \mathbf{X} is on the axis of rotation, it will satisfy $A\mathbf{X} = \mathbf{X}$, or equivalently, $(A - I)\mathbf{X} = \mathbf{0}$.

Exercise 1

Construct the matrix A as the product of the matrices $R_x(\frac{\pi}{6})$ and $R_y(\frac{\pi}{4})$.

Exercise 2

Find an \mathbf{X} on the axis of rotation by using the equation $(A - I)\mathbf{X} = \mathbf{0}$. Thus, \mathbf{X} is in the null space of $A - I$. Such an \mathbf{X} can be found by applying Maple's **nullspace** command to $A - I$, which Maple lets us form via the syntax $A - I$. The **nullspace** command returns a set of vectors, so \mathbf{X} will have to be extracted from this set.

Exercise 3

Plot the line segment from $-\mathbf{X}$ to \mathbf{X} . If we parametrize this line segment as $t\mathbf{X}$ we can plot it with the **spacecurve** command, letting t lie in the interval $[-1,1]$. Assign this plot to a variable so it can be used in a later exercise.

```
> tX := evalm(t*X);

> f1 := spacecurve(tX,t=-1..1, color=black, axes=boxed, scaling=constrained,
labels=[x,y,z], labelfont=[TIMES,BOLD,14]): f1;
```

Exercise 4

The plane P through the origin perpendicular to \mathbf{X} is called the "plane of rotation." Since this plane contains the origin, it is a subspace of R^3 . If the vector \mathbf{X} is converted to a 1×3 matrix, Maple's **nullspace** command will produce a basis for the plane P (the null space of the matrix-form of \mathbf{X}). Why? Obtain this basis, naming its elements **N1** and **N2**. Plot line segments through **N1** and **N2** as was done in Exercise 3. Join this plot with the one from Exercise 3 with **display3d** from the *plots* package, and assign the merged graphs to a variable for use later in Exercise 4. Be sure to use a 1-1 aspect ratio so that orthogonal vectors appear orthogonal.

The conversion of \mathbf{X} to a matrix is accomplished by Maple's **convert** command, with "matrix" as the parameter. This will be a 3×1 matrix which the **nullspace** command will reject. Apply the **transpose** operator to produce a 1×3 matrix to give to the **nullspace** command.

Maple's **nullspace** command does not yield normalized vectors. After obtaining the basis of the null space, normalize the vectors with Maple's **normalize** command.

Show that the basis vectors are not necessarily orthogonal to each other. They are orthogonal to \mathbf{X} . Use Maple's **dotprod** command to compute the dot products of vectors.

Exercise 5

The expectation should be that multiplication by A rotates elements within the plane orthogonal to the axis of rotation. To test this hypothesis, multiply both **N1** and **N2** by A . Then use Maple's **angle** command to find the angle between **N1** and $A \mathbf{N1}$, and between **N2** and $A \mathbf{N2}$.

A second multiplication by A should rotate $A \mathbf{N1}$ and $A \mathbf{N2}$ by the same amount. Verify this.

Exercise 6

Continue to explore, by visual means, the idea of rotations in the plane P . Create a plot of 15 successive applications of the matrix A to the basis vectors found in Exercise 2. If each application of A rotates these basis vectors through a fixed angle, and if they remain in P , their collective image should "show" the plane P .

Since working with exact expressions can lead to memory-consuming "expression swell," it is wise to convert the computations to the numeric

domain. Map the **convert** operator, with the *float* option, onto the matrix **A** and the vectors **N1** and **N2**, coining new names for these numeric versions. For example, we might call the floating point versions of these quantities **B**, **NN1**, and **NN2**, respectively.

Multiplication of **NN1** by **B**, *k*-times, produces $B^k \mathbf{NN1}$. We can produce the desired plots using Maple's `seq` command as follows. Finally, use **display3d** to merge these plots with the plot from Exercise 4.

```
> s1 := seq(evalm(t*((B^k) &* NN1)),k=1..15): s2 := seq(evalm(t*((B^k)
&* NN2)),k=1..15):
> f4 := spacecurve({s1},t=0..1,color=blue): f5 := spacecurve({s2},t=0..1,color=green):
```

Exercise 7

Determine, in the plane **P**, the angle of rotation caused by multiplication by **A**.

Use Maple's `angle` command, but express the answer in degrees, as a floating point number.

4.5 On Line

Restart Maple to clear its memory of all defined variables, and then re-initialize by loading the *linalg* and *plots* packages.

Exercise 1

Imagine that you are an astronomer who is investigating the orbit of a newly discovered asteroid. You want to determine (a) what is the closest the asteroid will come to the sun and (b) what is the furthest away from the sun the asteroid will get. To solve your problem, you will make use of the following facts.

a) Asteroids have orbits which are approximately elliptical, with the sun as one focus.

b) In polar coordinates an ellipse with one focus at the origin can be described by a formula of the form

$$r = \frac{c}{1+a \sin(\theta)+b \cos(\theta)}$$

where a , b , and c are constants.

You have also collected the data below; where r is the distance from the sun in millions of miles, and θ is the angle between the vector from the sun to the asteroid and a fixed axis through the sun. The data is, of course, subject to experimental error.

$$\begin{bmatrix} \theta & 0 & .6 & 1.8 & 1.4 & 2.1 & 3.2 & 5.4 \\ r & 329.27 & 313.8 & 319.49 & 310.91 & 327.88 & 374.91 & 367.49 \end{bmatrix}$$

Your strategy is to use the given data to find values of a , b , and c which make the formula for the ellipse to agree as closely as possible with the given data.. This will involve setting up a system of linear equations in a , b , and c and solving the normal equation. (Give the augmented matrices for both the original system and the normal equation.) You will then graph the given formula and measure the desired data from the graph.

Note: Once you have found values of a , b , and c , you will need to plot the orbit of the asteroid.

This can be done with the following command where r is the function that defines the ellipse.

```
> p3 := plot([r,t,t=0..2*Pi],coords=polar):
```

Remark: As stated, this is an inherently nonlinear problem which we solve using linear equations. There are more accurate techniques based on multivariable calculus. These techniques are also considerably more complicated than our solution.

Exercise 2

Maple has a `leastsqrs` command from the `linalg` package that is easier (and better) for solving least squares problems than simply solving the normal equations. Use this command to solve the over-determined linear system developed in Exercise 1. An appropriate syntax is as below where \mathbf{A} is the coefficient matrix for the system and \mathbf{F} is the vector of constants on the right side of the equations.

```
> leastsqrs(A,F);
```

Chapter 5

Determinants

Determinants are extremely useful in many contexts. You will, for example, use them constantly when you study eigenvalues and eigenvectors later in the text. In addition, you will see them used to write formulas for the solutions to many applied problems. In particular, determinants are used extensively in the study of differential equations and in the study of advanced calculus. Determinants are also used extensively in studying the mathematical foundations of linear algebra. Computers, however, do not generally use determinants for computations. Much faster and more efficient numerical techniques have been found. Thus, we will not provide any computer exercises for this chapter.

The reader should be aware, however, that Maple will compute determinants. The appropriate command is “`det(A)`”. Incidentally, Maple uses the methods of the next section to compute determinants rather than the methods already described.

Chapter 6

Eigenvectors

6.1 On Line

Restart Maple to clear its memory of all variables, and re-initialize it by loading the *linalg* package.

In the **On Line** section for Section 5.1 we commented that virtually anything you might use determinants for, a computer would do otherwise. This includes finding eigenvalues. Algorithms for computing eigenvalues are very sophisticated, and will not be described in this text. However, we will point out that the techniques do not involve finding the characteristic polynomial and determining its roots. In fact, the numeric recipes for finding eigenvalues are so good that they are often used to find roots of polynomials!. The exercises in this section explore this idea.

Exercise 1

If A is the matrix $\begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix}$, show (by hand) that the characteristic polynomial is $p(\lambda) = \lambda^2 + a\lambda + b$. Use this to construct a matrix A_1 which has $p_1(\lambda) = \lambda^2 + 7\lambda + 1$ as its characteristic polynomial. Check that your matrix has the correct characteristic polynomial using Maple's **charpoly** command.

Note: The **charpoly** command requires that you name the variable to be used in the characteristic polynomial. Thus, an appropriate syntax might be

```
> p_1:=charpoly(A,lambd);
```

Also, Maple uses $p(\lambda) = \det(\lambda I - A)p(\lambda) = \det(A - \lambda I)$ as the definition

of the characteristic polynomial. The relation between the two is that if A is $(n \times n)$, then the characteristic polynomial we compute in the text is $(-1)^n$

Use Maple's **eigenvals** command to compute the eigenvalues of $A1$ and hence the roots of $p_1(\lambda)$

Note: An appropriate syntax for the solve command is

```
> solve(p1 = 0, lambda);
```

Exercise 2

Compute the characteristic polynomial for the matrix $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -c & -b & -a \end{bmatrix}$.

Use this result to obtain the roots of the polynomial $p_1(\lambda) = \lambda^3 + 8\lambda^2 + 17\lambda + 10$. Test the roots by substitution back into $p(\lambda)$.

Exercise 3

Let $A1$ be the matrix obtained in Exercise 3. Obtain the eigenvectors of $A1$, normalizing them so the first element in each is 1. What do you then notice about these eigenvectors? Use the pattern you articulate to give a general prescription of the eigenvectors of an $(n \times n)$ matrix of the form A . Prove your answer.

The eigenvectors for $A1$ may be computed using the following command.

```
> q := eigenvecs(A2);
```

Note that there are three lists in q , and in each list there are three members. The first member of each list is the eigenvalue. The second member of each list is the algebraic multiplicity, the number of times that eigenvalue is a root of the characteristic equation. The third member of each list is a set of eigenvectors. Here, each such set contains a single eigenvector. Hence, these eigenvectors can be referenced as follows.

```
> v1 := q[1][3][1]; v2 := q[2][3][1]; v3 := q[3][3][1];
```

Exercise 4

Find a 4×4 matrix A whose characteristic polynomial is $p(\lambda) = \lambda^4 + 3\lambda^2 - 5\lambda + 7$. Obtain the roots of this polynomial by finding the eigenvalues of the matrix A . Check your result with Maple's **solve** command.

The matrix whose characteristic polynomial is $p(\lambda)$ is known in mathematics as the companion matrix. Maple has the built-in command **companion** for finding the companion matrix for a polynomial. Maple generates a companion matrix that is the transpose of what you might find in some differential equations texts.

```
> p := x^4 + 3*x^2 - 5*x + 7;
> A := transpose(companion(p,x));
```

Find the eigenvalues of A.

```
> eigenvals(A);
```

Maple has expressed the roots of the characteristic polynomial with its "RootOf" notation. This is a shorthand for what could be large and complex expressions for the exact value of the roots. There are several options available at this point. First, apply the **allvalues** command to the RootOf structure. This will return the eigenvalues as exact values, containing complicated expressions involving radicals.

```
> q := eigenvals(A);
> q1 := allvalues(q);
```

These expressions are too unwieldy to work with. Convert them to floating point numbers with the **evalf** command.

```
> evalf(q1);
```

Another alternative is to include at least one floating point number in the matrix A. When **eigenvals** sees the float, it will compute the eigenvalues numerically.

```
> A1 := map(convert,A,float);
> eigenvals(A1);
```

Finally, solve the equation $p(\lambda) = 0$ numerically by using the **fsolve** command. For polynomial equations, this command accepts the parameter "complex" to indicate that all roots, both real and complex, are to be found.

```
> fsolve(p,x,complex);
```

6.2 On Line

Restart Maple to clear its memory of all variables, then reinitialize by loading the *linalg* package.

Create your own eigenvalue problem by constructing a 3 x 3 matrix A with prescribed eigenvalues and eigenvectors. The eigenvalues are the diagonal elements in a diagonal matrix D , while the eigenvectors are the columns of a 3 x 3 matrix P .

A simple way to construct the eigenvector matrix P is as a random matrix. Hence, define f , a function which generates random integers in the interval $[-10,10]$.

```
> f := rand(-10..10):
```

Let P be a random 3 x 3 matrix with entries determined by f .

```
> P := randmatrix(3,3,entries=f);
```

Let the eigenvalues be 2, 2, and 3, in that order. Create a diagonal matrix with these elements on the diagonal, but assign the matrix to the name d , not D . The letter "D" in Maple is reserved for one form of the differentiation operator and Maple will not let you assign to it.

```
> d := diag(2,2,3);
```

The matrix $A = P D P^{(-1)}$ will have eigenvalues 2, 2, and 3 - in that order - and will have the columns of P as eigenvectors, in corresponding order.

```
> A := evalm(P &* d &* inverse(P));
```

Exercise 1

By computing $A \mathbf{X}$ for each column \mathbf{X} of P , verify that each column of P is an eigenvector of A . Columns of P can be referenced by Maple's `col` command. Clearly, $A \mathbf{X} = \lambda \mathbf{X}$ must hold for each eigenpair of eigenvector \mathbf{X} and eigenvalue λ .

Exercise 2

Verify that the diagonal elements of D are the eigenvalues of A by using Maple's `eigenvals` command to determine the eigenvalues of A directly from A itself. There is, however, no canonical ordering for the results of this command, so Maple need not order the eigenvalues as 2, 2, 3.

```
> eigenvals(A);
```

Exercise 3

By applying Maple's **eigenvects** command to A , again verify that the columns of P are the eigenvectors. The **eigenvects** command returns lists with three members in each list. These three members are first, the eigenvalue; second, the algebraic multiplicity of the eigenvalue - the number of times the eigenvalue was a root of the characteristic equation; and third, a set of eigenvectors associated with the eigenvalue in the list.

Extract the eigenvalues and eigenvectors by adroit use of the selector bracket notation $[\]$. (See Exercise 3 in the On Line exercises for Section 6.1.) Again, there is no canonical ordering for the lists produced, or for the eigenvectors associated with an eigenvalue of multiplicity greater than 1. Executing the **eigenvects** command on different occasions can result in a different ordering each time.

The eigenvectors computed by the **eigenvects** command may not "look like" the columns of P . The columns of P may be constant multiples of the corresponding vectors, or, in the case of multiple eigenvectors, the columns of P could simply be a different basis for the eigenspace associated with the repeated eigenvalue. Compare the third column of P with the eigenvector Maple found for the eigenvalue 3, determining any multiplicative factor needed to make the eigenvectors match exactly.

To show two bases $\{P_1, P_2\}$ and $\{v_1, v_2\}$ are equivalent, you need to show that linear combinations of one set of basis vectors yield the other basis vectors. Use the **rref** command on a matrix containing as its columns the first two columns of P and the eigenvectors Maple found for the eigenvalue 2. How will this show the equivalence of the bases?

Exercise 4

For the $n \times n$ matrix A , the characteristic polynomial $p(\lambda)$ has been defined in this text as $p(\lambda) = \det(A - \lambda I)$. Some texts use $\det(\lambda I - A)$, thereby making the two definitions differ by a factor of $(-1)^n$. Maple's built-in **charpoly** command for generating the characteristic polynomial uses the latter convention. The advantage of Maple's definition is that for an $(n \times n)$ matrix, the highest order term

In Maple, compare these two methods for obtaining the characteristic polynomial. The **charpoly** command takes as arguments the matrix A and a variable to be used in the output polynomial. Maple computes determinants via the **det** command, and also allows the syntax $A - \lambda$ as a short form of $A - \lambda I$. Finally, typing out the "name" of the Greek letter λ causes Maple to print that letter as a Greek letter.

```
> charpoly(A,lambda);
```

```
> det(A-lambda);
```

Note that the two polynomials are just negatives of each other.

There is only one degree three polynomial with roots 2, 2, and 3, that has λ^3 as its highest degree term. What is this polynomial? (Hint: Write it as a product of linear factors and then expand.)

Note that this is the characteristic polynomial of A as found by Maple.

6.3 On Line

Restart Maple to clear its memory of all defined variables, and re-initialize by loading the *linalg* package.

In these exercises complex numbers will appear. Maple uses the letter "I" for $\sqrt{-1}$, so that the complex number $z = 2 + 3i$ is entered into Maple as $z = 2 + 3*I$. It is also useful to remember that if $z = 2 + 3i$, then $\bar{z} = 2 - 3i$ is the complex conjugate of z . Thus, the complex conjugate of a real number x is that real number itself, since the imaginary part (the part with the i) is zero.

Maple's command for conjugating a number is **conjugate**. Its commands for extracting the real and imaginary parts of a complex number are **Re** and **Im**, respectively. In purely numeric contexts these commands usually need no additional boosts. In symbolic and exact contexts, these commands generally work only if an additional **evalc** (evaluate complex) is applied. Thus,

```
> Re(2 + 3*I); Im(2 + 3*I);
```

but

```
> Re(x + I*y); Im(x + I*y);
```

thereby requiring

```
> evalc(Re(x + I*y)); evalc(Im(x + I*y));
```

Exercise 1

Use Maple's **eigenvectors** command to obtain the eigenvalues and eigenvectors of the matrix $A = \begin{bmatrix} 1 & -3 \\ 1 & 1 \end{bmatrix}$ from Example 2 of section 5.3. Notice that the eigenvalues are complex conjugates, as are the eigenvectors as well.

Exercise 2

An $n \times n$ matrix A with complex entries is said to be **Hermitian** if the conjugate of the transpose equals A . Thus, A is Hermitian if $\overline{A^t} = A$. A moment's reflection reveals that the conjugate of the transpose equals the transpose of the conjugate, that is, $\overline{A^t} = \overline{A}^t$. Some texts denote the conjugate transpose of A by the symbol A^* so that $A^* = \overline{A}^t = \overline{A^t}$, and A is Hermitian provided $A = A^*$.

Give an example of a 3×3 Hermitian matrix containing as few real numbers as possible, and having no entries zero.

To verify that your matrix is Hermitian, you need to take both the transpose and the complex conjugate. Maple has the built-in **htranspose** command for this. Apply it to your matrix A , and then separately apply the **transpose** and **conjugate** commands. In Maple, the simplest way to do this is to map **conjugate** onto the transpose of A .

```
> htranspose(A);
> map(conjugate,transpose(A));
```

One remarkable property of Hermitian matrices is that their eigenvalues are real. Apply Maple's **eigenvals** command to your matrix A in an effort to verify the truth of this claim. You will most likely obtain large, complicated expressions for the symbolically exact eigenvalues. They might even contain the symbol "I", making it look like the eigenvalues are complex! To determine if these eigenvalues are real, you want to show that for each, the imaginary part is zero.

After obtaining the exact eigenvalues, convert them to floating point numbers by using the **evalf** command. Since **eigenvals** returns a sequence of eigenvalues, you may need to convert this return to a list (use [...]) before the **evalf** command works. The conversion to decimals of any radicals in your eigenvalues may yield expressions with very small imaginary parts. You can instruct Maple to truncate these small numbers to zero by using the **fnormal** command.

Instruct Maple to extract and simplify the imaginary part of each eigenvalue. These imaginary parts should reduce exactly to zero. Remember to use both **evalc** and **Im**, as well as **simplify**, on each exact eigenvalue.

Obtain the exact real part of each eigenvalue. These expressions will be real, but complicated. The point of the activity is for you to realize that exact values for the roots of cubic equation are unpleasant expressions to work with. Just because Maple is able to provide the roots exactly does not mean that these expressions are always useful or simple.

Finally, apply **evalf** directly to the exact real parts of the eigenvalues.

Compare the values to what you got when floating the unsimplified "complex" version of the eigenvalue.

Exercise 3

Change one of the entries of A from Exercise 2, making A non-Hermitian. Then, recalculate the eigenvalues. Are they again real?

Make the change to A in Maple by creating a matrix B via substitution of a new value into the matrix A. Reference an element of A, say the 1,1-element, by A[1,1], and make a substitution of a new value for such an element into op(A), not just into A. Choosing the new value to be a floating point number will mean that **eigenvals** will return floating point numbers directly. Since Exercise 2 made a thorough study of the complexity of exact calculations, work numerically in this exercise.

Exercise 4

For the matrix A of Exercise 2, compare the action of transpose and htranspose. Is there any difference if these operators are applied to matrices with just real entries?

Exercise 5

In Maple, let A be a real, symmetric 3 x 3 matrix with as many of its entries as possible distinct. Obtain $B = I + iA$, where I is the 3 x 3 identity matrix and $i = \sqrt{-1}$. Let $C = I + \frac{B^{(-1)}}{2}$ and compute both $C C^*$ and $C^* C$. Can you prove that what you observe is always true? (Hint: Begin with the equality $B + B^* = 2I$ and multiply by $B^{(-1)}$ on the left and by $[B^*]^{(-1)}$ on the right.)

Notes: To generate a random symmetric matrix in Maple, add the parameter *symmetric* to the **randmatrix** command.

```
> A := randmatrix(3,3,symmetric);
```

The matrix B can be obtained in Maple if due note is taken of Maple's usage of I for the imaginary unit $\sqrt{-1}$, and due care is taken to distinguish between the written symbols I and i, standing respectively for the identity matrix and the imaginary unit $\sqrt{-1}$.

Since an identity matrix is a diagonal matrix with just 1's on the diagonal, the **diag** command can be used to create an identity.

```
> B := evalm(diag(1$3)+I*A);
```

The matrix C likewise requires use of an identity matrix.

```
> C := evalm(diag(1$3)+inverse(B)/2);
```

6.5 On Line

Restart Maple to clear its memory of all variables, then re-initialize by loading the *linalg*, *plots*, and *plottools* packages.

```
> restart;
> with(linalg): with(plots): with(plottools):
```

The purpose of this exercise set is to explore the relationship between eigenvalues, eigenvectors and the geometry of quadratic forms.

Exercise 1

Use Maple's **implicitplot** command from the *plots* package to obtain a graph of the ellipse defined by the quadratic equation $x^2 + 4y^2 = 1$. Be sure to use a 1-1 aspect ratio so that there is no distortion in the scaling. Then, obtain A, the matrix of the quadratic form defined by this same equation. This can be done by typing in A, by clever use of partial differentiation, or by Maple's **hessian** command. Obtain the eigenvalues and eigenvectors of A.

```
> q := x^2 + 4*y^2 = 1;
> implicitplot(q,x=-1..1,y=-1..1,scaling=constrained);
```

By inspection, we can write $A = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$. Alternatively, we can note

that $A = \frac{1}{2} \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$, where $f(x,y)$ is the left hand side of the defining quadratic equation, and subscripts denote partial derivatives. Thus,

```
> f := lhs(q);
> fxx := diff(f,x,x); fxy := diff(f,x,y); fyx := diff(f,y,x);
fyy := diff(f,y,y);
```

The array of second partial derivatives of the form $\begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$ is called the hessian matrix, and is returned in Maple by the **hessian** command.

```
> hessian(g(x,y),[x,y]);
```

Hence, the matrix of a quadratic form is simply one-half the hessian matrix. The simplest way to get the 1/2 into the hessian is by including it in the quadratic function. Else, an **evalm** is needed to multiply the hessian matrix by that 1/2.

```
> A := hessian(lhs(q)/2, [x,y]);
```

The eigenvalues and eigenvectors can be obtained by use of the **eigenvects** command.

We next seek to relate the eigenvalues and eigenvectors to the lengths of the semi-major and semi-minor axes. From the graph, these axes are 1 and 1/2 respectively. The eigenvalues are 1 and 4, respectively for eigenvectors that have the directions of the ellipse's axes. Hence, the scale factors by which to multiply the eigenvectors to get the semi-major and semi-minor axes are $\frac{1}{\sqrt{\lambda_k}}$, where $k = 1, 2$.

Exercise 2

Using Maple's **implicitplot** command, obtain a graph of the function defined implicitly by the quadratic equation $2y^2 + xy + x^2 = 1$. Be sure to use an aspect ratio scaled to 1-1. It may also be edifying to increase the number of points used in the plot by adding the parameter *numpoints* = 1000 to the **implicitplot** command. Assign the plot to a variable so that it can be re-used in subsequent exercises.

Exercise 3

For the quadratic equation in Exercise 2, form the matrix A of the quadratic form defined by the equation. Use Maple's **hessian** command, and check the result by taking partial derivatives of the left hand side of the defining quadratic function. Obtain the eigenvalues and eigenvectors of A via Maple's **eigenvects** command. Extract, and give unique names to, the eigenvalues and eigenvectors. Obtain floating point approximations for the eigenvalues, and normalize the eigenvectors to have length 1 with Maple's **normalize** command. Apply Maple's **radsimp** command to the 2-norm (computed via **norm**) of each normalized eigenvector to verify that each indeed has length 1. Use the **arrow** command from the *plottools* package to superimpose the normalized eigenvectors on the graph of the ellipse. Define each arrow separately. The arrow command takes five parameters: a list of coordinates for the tail, here the origin; a list of coordinates for the point, here the entries in the normalized eigenvectors; then three sizing parameters which experiment shows are well chosen as .05, .1, and .05. Color each

arrow differently. Then use the **display** command from the *plots* package to merge the graph of the ellipse and the two arrows into one graph. To convert a vector to a list, use the **convert** command with option *list*. Thus, if the eigenvectors are named V1 and V2, you might enter

```
> a1:=arrow([0,0],convert(V1,list),.05,.1,.05, color=green):
> a2:=arrow([0,0],convert(V2,list),.05,.1,.05, color=blue):
> display([p1,a1,a2],scaling=constrained);
```

Print a copy of this final graph of the ellipse with the normalized eigenvectors.

Exercise 4

On the plot printed in Exercise 3, draw in the axes determined by the eigenvectors. On these axes put tick marks every quarter unit, noting that each eigenvector is one unit long. Use a ruler to guarantee the accuracy of your tick marks. According to the general theory, the ellipse should cross the new axes at points whose coordinates in the system determined by the eigenvectors are

$(\frac{1}{\sqrt{\lambda_1}}, 0)$, $(-\frac{1}{\sqrt{\lambda_1}}, 0)$, $(\frac{1}{\sqrt{\lambda_2}}, 0)$, $(-\frac{1}{\sqrt{\lambda_2}}, 0)$, where λ_1 and λ_2 are the eigenvalues of A. Verify this by estimating the appropriate coordinates from your graph.

Exercise 5

Verify in general that the scale factors $\frac{1}{\sqrt{\lambda_k}}$, $k = 1, 2$, indeed convert normalized eigenvectors into vectors of precisely the length of the semi-major and semi-minor axes. Begin by obtaining floating point values of $\frac{1}{\sqrt{\lambda_k}}$, $k = 1, 2$. Then scale the normalized eigenvectors by these factors. Next, use Maple's arrow command to build a plot of the ellipse, and the newly scaled eigenvectors. A plot of the ellipse, and these new basis vectors should show that with this scaling the vectors coincide precisely with the semi-major and semi-minor axes.

Exercise 6

Find the equation of an ellipse, centered at the origin, and for which the major axis is 4 units long and lies along the line determined by the vector $v_1 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, and for which the minor axis is 2 units long. (Hint: If you

can figure out the eigenvalues and eigenvectors, you then can find matrices Q and D for which $A = Q D Q^t$, where A is the (symmetric) matrix for the quadratic form corresponding to the ellipse.) Graph the ellipse and the scaled eigenvectors for A , demonstrating that the proper scaling of the eigenvectors gives them the lengths of the semi-major and semi-minor axes.

Since the axes of the ellipse are orthogonal, you need to get a vector orthogonal to the vector v_1 . This can be done in the plane by interchanging the x - and y -coordinates, and negating one component of the resulting vector. Even a casual inspection reveals why this works.