Introduction to Scilab

Graeme Chandler
Mathematics Department
The University of Queensland

Stephen Roberts
Department of Mathematics
Australian National University

August 7, 2002

Contents

1 Introduction.

High level scientific computing environments such as Matlab, Rlab, Octave and Scilab are an enjoyable way to solve problems numerically. The computational problems arising in most undergraduate courses can be solved much more quickly using these environments, than with the standard programming languages (Fortran, indexFortran C, Java, etc.). It is particularly easy to generate some results, draw graphs to look at the interesting features, and then explore the problem further. By minimizing human time, it is particularly useful in the initial investigation of real problems; even though they may eventually have to be solved using more computationally efficient ways on super computers.

Matlab is by far the most popular of these environments, it has a large user base and many resources are available on the Web. Indeed a version of this document "An Introduction to Matlab" is available on the web at http://www.maths.uq.edu.au/~gac/mlb/mlb.html.

The other three packages are public domain or open source packages. In this document we concentrate on Scilab. Scilab provides an environment which uses matrices (both full and sparse) as it basic data type. Algorithms are available for the numerical solution of differential equations, optimisation problems, interpolation and quadrature. High quality fortran and C codes can be easily linked into Scilab. The package provides excellent 2 and 3 dimensional graphics which can be readily incorporated into reports and publications. Simple user interfaces can be built using Scilab. The package is also available for all standard machines, PC's, Macs, LINUX and Unix.

All these attributes (and more) lead us to use Scilab as the basic package for our computational science courses at the Australian National University. Indeed the ability for students to have access to the software on their home machines is the main reason for a choice of open software. We feel that Scilab provides the best compromise of free software and a sophisticated range of builtin facilities.

1.1 Learning Scilab

This introduction gives a quick way to become familiar with the most important parts of Scilab. The first five sections emphasize simple arithmetic, matrix-vector operations (including solving systems of equations), and graphing functions and data. The later sections describe some more advanced features, including 3D graphics. There are also some suggestions about using Scilab to do larger projects, and including Scilab results and graphs in reports.

The best way to used this introduction is to sit down at a computer and type in the commands as they are described. Look at Scilab's response, and check that the answers are what you expect. It is also a good idea to do the small exercises. It makes sure that the commands become part of an active Scilab vocabulary. Each lesson should take less than one hour. More information about any Scilab command can be found by using the on line help features described in lesson 3.

These notes assume basic familiarity with the Windows interface. For instance, you need to know about

- cutting and pasting within and between windows,
- editing files, and
- moving between windows and resizing them.

If you are unsure about this, seek help from another student or tutor.

1.2 Further References

Our Scilab Resource Page, http://www.maths.anu.edu.au/~steve/Scilab, is a good place to start. It provides links to documentation, other Scilab web pages and general information.

1.3 Starting Scilab

The Scilab commands are the same on all machines.

1. First sit down at a PC. If necessary, close any programs left running by a previous user. The simplest way is to use the Windows 'Start' button then the 'Shut Down' option.
2. When prompted, use the [CTRL]-[ALT]-[DEL] combination to bring up the logon screen. (You will be asked for a name and password, but ignore this and just press the [ENTER] key.)
3. You should now see the standard Windows desktop with a screen of icons.
4. Find the icon labelled 'Scilab' and double click on it. After another pause, the Scilab logo appears briefly, then the 'Scilab Command Window' remains on the screen. It ends with the words:-

```
           ===========
           S c i l a b
           ===========


                scilab-2.6
        Copyright (C) 1989-2001 INRIA


Startup execution:
  loading initial environment

-->
```

Scilab is now ready!

1.4 Typing Commands

All commands to Scilab are typed in after the Scilab prompt, i.e. the symbol `-->` . They are then sent to Scilab to be implemented by pressing the [ENTER] key.

- Any typing error can be corrected before the [ENTER] key is pressed.
  - Use the keypad left and right arrows or the mouse to move to the error.
  - Use the [DEL] key to delete the mistakes, and then type in the correction.
  - When the line is correct, just press [ENTER] to send the command to Scilab. (It is not necessary to go to the end of the line.)

Usually errors are not noticed until Scilab displays an error message. However *it is not necessary to retype the whole command.*

- Previous commands can be corrected and reused to save typing.
  - Just press the keypad up arrow and the previous command appears.
  - Make the necessary corrections and press [ENTER] to run the corrected command.

Now go on to the first lesson. Type in all the commands as they are shown, and make sure the Scilab response is what you expect.

2 Simple Calculations

2.1 Basic Arithmetic.

The simplest way to start with Scilab is to use it for simple calculations. Scilab has a wide range of functions and is able to use complex numbers as well as reals. The following simple commands have obvious meanings. Type them in and see what happens.

In the examples below, all the text after the // is just a comment or an explanation. You do not have to type it in. It would just be ignored by Scilab.

```
                // The words after '//' are comments
                // and explanations.  Do not type them in.
(-1+2+3)*5 - 2/3        // The four arithmetic operations
2^3                     // Means 2 to the power 3
%pi                     // The mathematical constant pi
exp( sin( %pi/2 ) )     // The usual functions are provided
                        // log, log10, cos, tan, asin, ...
%e                      // The mathematical constant e
```

Like a calculator, Scilab does all calculations correct only to about 16 significant decimal digits. This is enough accuracy for most purposes. For convenience, usually only the first 5 significant digits are displayed on the screen. Some more examples.

```
%pi
22/7                  // pi is not the same as 22/7!
11*(15/11) - 15       // This shows there is roundoff error
                      //   when Scilab uses fractions.
cos( %pi/3 )          // These are familiar trig functions.
sin( %pi/6 )          // Note Scilab always uses radians.
```

Scilab also uses variables to store intermediate answers. A variable can have almost any name, but it must begin with a letter. Scilab distinguishes between upper and lower case letters.

```
x = 2+3
y = 4+5
result1 = x/y
```

Sometimes the values of intermediate calculations are not needed and we do not want them to be displayed. If a semicolon (;) is placed after a command, the results are not displayed.

```
p = 2+3 ;                    // The semicolons supress
q = 3+5 ;                    // unwanted output.
ratio = p/q
```

A number of commands can be placed on the one line, provided they are separated by a comma (,) or a semicolon (;). (Use a semicolon, unless you want the answer displayed.)

```
p = 2+3 ; q2 = x + 4 , ratio2 = p/q2       // All on one line
```

Parentheses can be used to make expressions clearer. Thus the last calculation could have been written as

```
ratio  =  (2+3)/(x+4)
```

Exercise 1 *In this exercise, save retyping by using the up arrow to edit the previous commands! Remember that multiplication is done with a '∗', it is not enough to put numbers next to each other.*

1. *Use Scilab to evaluate* $\log(s^2 - 2\,s\,\cos(\pi/5) + 1)$ *where* $s = .5$.
2. *Now use Scilab to evaluate* $\log(s^2 - 2\,s\,\cos(\pi/5) + 1)$ *where* $s = .95$.
3. *Finally, evaluate* $\log(s^2 - 2\,s\,\sin(\pi/5) + 1)$ *when* $s = 1$.

2.2 Complex Numbers.

Scilab handles complex numbers as easily as real numbers. When you start Scilab the variable %i stands for $\sqrt{-1}$.

```
x  =  2 + 3*%i  ,  y  = 1 - 1*%i
z1 = x - y ,  z2 = x * y ,  z3 = x / y
abs( x )
real( x )
imag( x )
atan( imag(x) , real(x) )  // The argument of a complex number

sin(x)                     // Scilab quickly calculates the
                           // sin  of a complex number.
```

Scilab's facility with complex numbers is handy, as using complex numbers often involves complicated arithmetic. Indeed, as the previous example shows, Scilab will effortlessly work out functions of complex numbers that are difficult to do from first principles. But be careful about the name used for $\sqrt{-1}$. When Scilab starts, the variable `%i` contains $\sqrt{-1}$. It is sensible to avoid using variables with names that start with `%`.

Here are two more complex calculations. Scilab can be used to demonstrate one of the most important relations in Mathematics: that $e^{\pi i} = -1$. (At least to within round off error!).

```
exp( %pi*%i ) + 1
%i^%i
```

Exercise 2

1. *The last command showed that according to Scilab, i^i is a real number. Can you explain mathematically why this is this so? (Hint: Use $i = e^{\pi i/2}$.)*

2. *If $y = i^i$, what are i^y and y^i? Can you work this out without Scilab?*

3 Help in Scilab

3.1 The Help Command

The help command is the easiest way to find out more about specific Scilab commands. If you have forgotten small details, for example. The command `help <name>` gives information about the Scilab command <name>.

```
help sin     // Information about sin.
help +       // Gives links to help on Scilab operator names
help log     // This is enough information about log
             //     to show  log  means log to the base  e .
```

Note that often the help information provides an example of how the command is used. This can be particularly useful when experimenting with a new command. You can use cutting and pasting to run the example commands.

In Scilab the `apropos` command can also be used to search for relevant information.

```
apropos logarithm  // Provides a list of
                   // functions related to logarithms
```

3.2 The Help Window

It is also possible to get help by clicking on the HELP menu above the command window. From the HELP menu, select the HELP DIALOG item and the list of help topics is displayed. The advantage of this method is that it is possible to navigate around different topics and zero in on useful commands. For example, in the help dialog window click on the chapter 'Linear Algebra'. Then find the item 'linsolve'. Once found you click on 'show' to see the relevant help page. Note that double clicking doesn't work in windows Scilab version 2.6 and there is a slight bug when you change chapters and scroll to new items. Choosing the item a second time seems to work.

Generally the help window is a good way to explore Scilab's commands.

3.3 Help on the Web

Have a look at our Scilab resources page, to get links to some manuals and user guides.

Exercise 3

1. *Is the inverse sine function one of Scilab's elementary functions? (The inverse sine is also known as* $\sin^{-1}$*, arcsin, or asin.)*
   (a) *How do you find* $\sin^{-1}(.5)$ *in Scilab? Use help to find out.*
   (b) *If* $x = .5$*, is* $\sin(\sin^{-1}(x)) - x$ *exactly zero in Scilab?*
   (c) *If* $\theta = \pi/3$*, is* $\sin^{-1}(\sin(\theta)) - \theta$ *exactly zeros in Scilab? What about* $\theta = 5\pi/11$*?*
2. *Does Scilab have a function to convert numbers to base 16, i.e. to hexadecimal form? (Hint: Use* `apropos` *to find a way to* ***convert*** *a* ***decimal*** *number to* ***hexadecimal****.) What is 61453 in base 16? Computers almost always represent numbers internally as hexadecimals.*
3. *Look for information about logarithms. Note that searching for logarithms fails where as logarithm succeeds. There are 8 entries, including the command,* `logm`*, for calculating the log* ***of a matrix!***
4. *To wind down, use the Scilab menu command* DEMOS*. This brings a menu of demonstrations and examples to explore.*
   (a) *Visit the graphics to see a number of attractive images.*
   (b) *I also like the car and trailer parking demo.*

4 Plotting Lines and Data

This section shows how to produce simple plots of lines and data.

Suppose we wish to plot some points. For example we are given the following table of experimental results.

x_k	.5	.7	.9	1.3	1.7	1.8
y_k	.1	.2	.75	1.5	2.1	2.4

To work with the data in Scilab set up two column vectors `x` and `y`.

```
x = [.5 .7 .9 1.3 1.7 1.8 ]'
y = [.1 .2 .75 1.5 2.1 2.4 ]'
```

(Vectors are discussed in detail in the next lesson; but we can use them to draw graphs without knowing all the details.) To graph y against x use the plot command.

```
plot2d(x,y, style=-1)
```

The graph should now appear. (If not, it may be hidden behind other windows. Click on the icon 'Figure No. 1' on the Windows task bar to bring the graph to the front.)

This graph marks the points with an '+'. Other types of points can be used by changing the `style=-1` in the plot2d command. (Use `help plot2d` to find out the details.)

Exercise 4 *Experiment with different values for the style. Negative values give markers, positive values coloured lines. As you experiment you should erase the previous plot. Use the command*

```
xbasc()
```

Commands starting with `x` *generally are associated with graphics commands (This comes from the X window system used on Unix machines). So* `xbasc()`*, is a* ***bas****ic graphi****x*** *command which* ***c****lears the graphics window.*

From the graph it is clear that the data is approximately linear, whereas this is not so obvious just from the numbers. Good graphs quickly show what is going on!

When you have finished looking at the graph, just click on any visible part of the command window, or on the Scilab icon on the task bar. More commands can then be typed in.

Exercise 5

1. *Plot the above data with the points shown as circles.*
2. *Plot the above data with the points joined by lines (use* `help plot`*).*
3. *Plot the points as green stars.*
4. *Find out how to add a title to the plot. (Hint: Use the command* `help plot`*. At the end of the help output there is a list of related commands. Find out about one of these with another use of the help command.)*

4.1 Adding a Line

From the graph it is clear that the points almost lie on a straight line. Perhaps the points are off the line because of experimental errors. A course in statistics will show how to calculate a 'line of best fit' for the data. But even without statistics, the line between the points $(.5, 0)$ and $(2, 3)$ is a good candidate for 'a line fitting the data'. Lets add this line to the plot and see how well it approximates the data. We do this by asking Scilab to plot the points $(.5, 0)$ and $(2, 2.7)$ joined by a line.

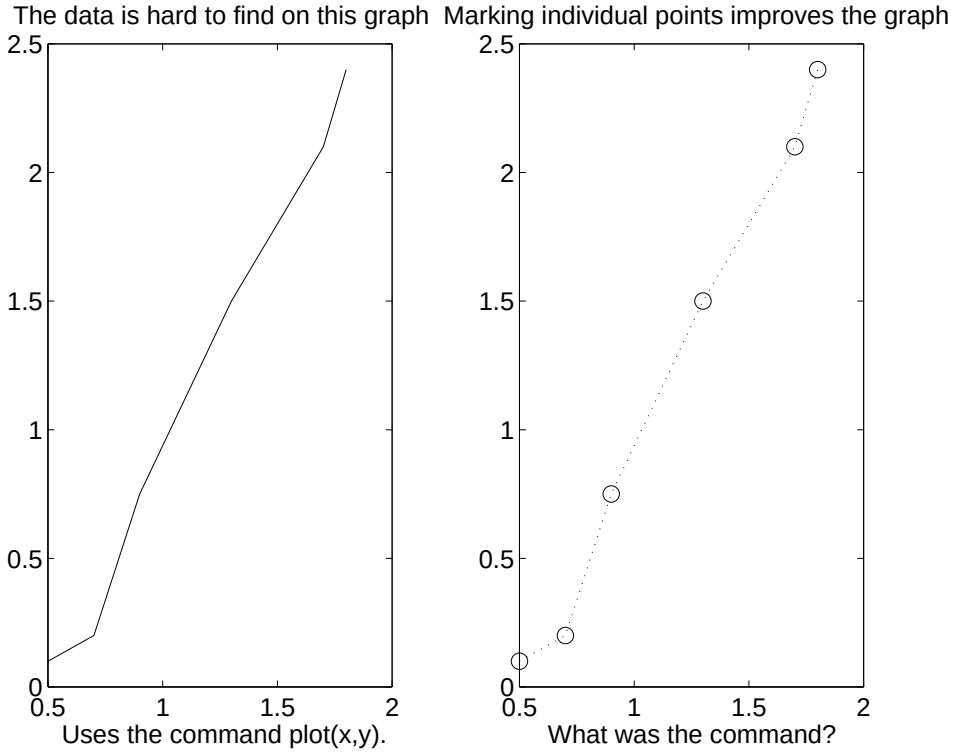
```
x_vals = [ .5 2]'        //  The X-coords of the endpoints.
y_vals = [ 0 2.7 ]'      //  The Y-coordinates
plot2d( x_vals,y_vals, leg='Line of Best Fit' )
xtitle( 'Data Analysis')
```

Note that `x_vals` contains the x–coordinates, `y_vals` contains the y–coordinates, and the two points are joined by a line because we don't specify a style (default style is a line).

The line of best fit can be found using the `datafit` function.

4.2 Hints for Good Graphs

There are many opinions on what makes a good graph. From the scientific viewpoint a simple test is to see whether we can recover the original data easily from the graph. A graph is supposed to add something, not remove information. For example if our data runs from 1 to 1.5 it is a bad idea to use an axis running from 0 to 10, as we will not be able to see the differences between the values. We give two more subtle recommendations..



4.2.1 Plot data as points

Often people use the very simplest command to plot data and automatically type just

```
plot2d( x, y )          // The simplest plot command
```

This simple command does not present the data here very well. It is hard to see how many points were in the original data. It is really better to plot just the points, as the lines between points have no significance; they just help us follow the set of measurements if there are several data sets on the one graph. If we insist on joining the points it is important to mark the individual points as well

Exercise 6 *Change the above plot command to show the data more clearly by plotting the data points as small circles joined by dotted lines. (Hint: There should be a third argument in the plot command containing the symbols for a line of dots and the marker type circles; and perhaps a colour symbol as well!). Use* `help plot2d` *to see the possible arguments.*

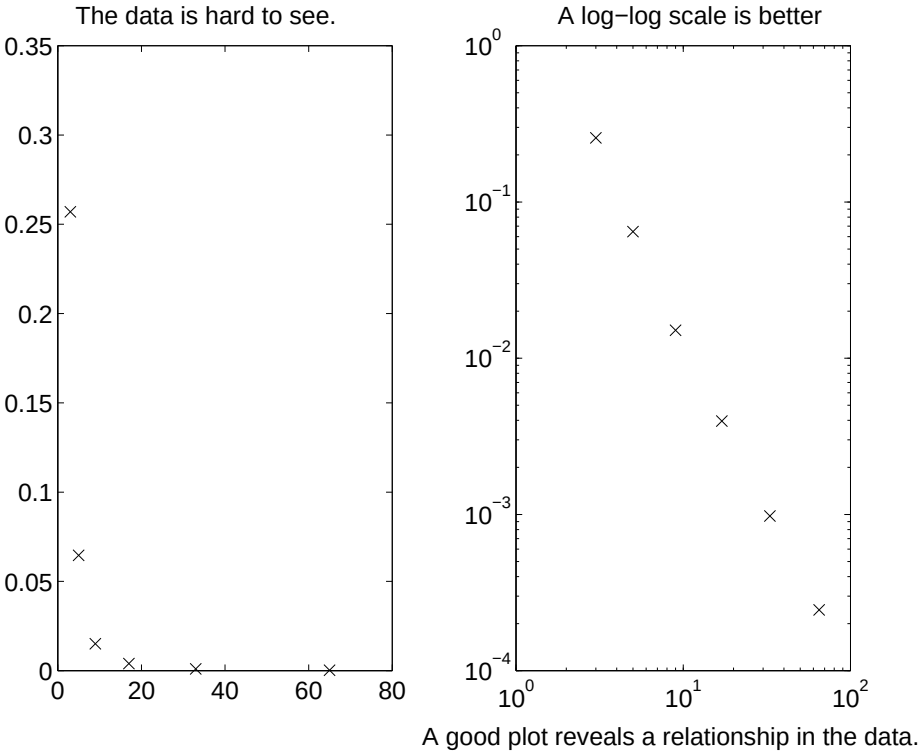
4.2.2 Choose a good scale

In the example in this section, it was easy to see the relationship between x and y from the simple plot of x against y. In more complicated situations, it may be necessary to use different scales to show the data more clearly. Consider the following model results

n	3	5	9	17	33	65
s_n	.257	.0646	.0151	3.96×10^{-3}	9.78×10^{-4}	2.45×10^{-4}

.

A plot of n against s_n directly shows no obvious pattern. (Note the dots, '...', in the next example mean the current line continues on the next line. Do not type these dots if you want to put the whole command on the one line.




```
n = [ 3 5 9 17 33 65 ]';
s = [ 2.57e-1  6.46e-2  1.51e-2 ...
            3.96e-3 9.78e-4  2.45e-4 ]' ;

plot2d( n, s, style=-1 )     // This is a poor plot!!
```

In fact it is hard to read the values of s_n back from the graph. However a plot of $\log(n)$ against $\log(s_n)$ is much clearer. To produce a plot on a log scale we can use either of the commands

```
 plot2d( log10(n), log10(s), style=-1)        // Two good plots
plot2d( n, s, style=-1 , logflag = 'll')     // using log scales!
```

It reveals there is almost a linear relationship between the logs of these two quantities.

Exercise 7 *1. Using* `help plot2d` *find a command which will use a log scale only for the s_n data. Is this better or worse than the log-log plot?*

2. Add a title to the last plot above and label the X and Y axes.

3. (A hard exercise.) After allowing for some experimental error, there is a simple relationship between n and s_n. Can you find a mathematical formula for this?

4. What would be the value of s_n when $n = 129$?

5 Matrices and Vectors

Although Scilab is a useful calculator, its main power is that it gives a simple way of working with matrices and vectors. Indeed we have already seen how vectors are used in graphs.

Remember to keep typing in the commands as they appear here, and observe and understand the Scilab response. If you mistype, it is easy to correct using the arrows and the [Del] key. Try to use the help facility to find out about unfamiliar commands. Otherwise ask another student or a tutor.

5.1 Solving Equations.

Most people will have seen systems of equations from school. For example, we may need to find x_1, x_2, and x_3 so that

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 1 \\ -2x_1 - 6x_2 + 4x_3 &= -2 \\ -x_1 - 3x_2 + 3x_3 &= 1 \end{aligned}$$

Although these problems can be solved manually by eliminating unknowns, this is unpleasant. Besides errors usually occur.

In first year Mathematics the problem is rewritten in matrix-vector notation. We introduce a matrix A and a vector b by

$$A = \begin{bmatrix} 1 & 2 & -1 \\ -2 & -6 & 4 \\ -1 & -3 & 3 \end{bmatrix}, \; b = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}.$$

Now we want to find the solution vector $x = [x_1, x_2, x_3]$ so that

$$Ax = b.$$

In spite of the new notation, it is still just as unpleasant to find the solution. In Scilab, we can set up the equations and find the solution x using simple commands.

```
                                        // Set up a system
A = [ 1 2 -1; -2 -6 4 ; -1 -3 3 ]       // of equations.
b = [ 1; -2; 1 ]
x = A\b                                 // Find x with A x = b.
```

Scilab should give the solution

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}.$$

A sound idea from manual computations is to substitute the computed solution back into the system, and make sure all equations are indeed satisfied. In Scilab we can do this by checking that the matrix vector product Ax gives the vector b, or better still, we check that $B - Ax$ is exactly the zero vector.

```
A*x          // Check that  Ax  and  b  are the same.
b
b - A*x ;    // So that we do not miss small differences,
             //  it is better to check that  b - A x = 0 .
```

The vector $b - Ax$ is known as the *residual*.

Exercise 8

1. *Change the middle element of A from -6 to 5 (i.e.* $a_{22} = 5$.*) What is the solution to this new system? What is the new residual? Why is the residual not exactly 0? (Hint: In Scilab the number* 1.23×10^{-5} *is displayed as* `1.23e-05`*. The column vector* $[1.23 \times 10^{-5}; 4.44 \times 10^{-6}]$ *could be displayed as*

```
1.0e-004 *
                          1.2300e-05
    0.1234     or         4.4400e-06
    0.0444
```

The user can control which form is used by using the command `format`. *Essentially the full accuracy can be seen using* `format('e',20)`.

2. *Suppose the middle element is changed from -6 to -5. Can Scilab solve this third system? Explain what has happened.*

5.2 Matrices and Vectors.

Much scientific computation involves solving very large systems of equations with many millions of unknowns. This section gives practice with the commands that are needed to work larger matrices.

The following code sets up a 4×4 matrix, A, and then finds some of its important properties. When `[ ]` is used to set up matrices, either blanks or commas (`,`) can be used to separate entries in a row. Semicolons (`;`) are used to begin a new row.

```
A =  [1 2 3 4 ; 1 4 9 16 ; 1 8 27 64 ; 1 16 81 256 ]
A'
det(A)
spec(A)
[D, X] = bdiag(A)
inv(A)
```

Even in the simple 4×4 case, it is tedious to evaluate determinants and inverses. But in Scilab they can be quickly calculated and used.

Indices can be used to show parts of a larger matrix. For example try

```
A(2,3),  A(1:2,2:4),   A(:,2),  A(3,:),   A(2:$,$)
```

In general `A( i:j, k:l )` means the square sub–block of `A` between rows i to j and columns k to l . The ranges can be replaced by just ':' if all rows or columns are to be included. The last row or column is designated `$`.

Exercise 9

1. *How would you display the bottom left 2×3 corner of A? How would you find the determinant of the upper left 3×3 block of A?*

2. *Find the Scilab command to calculate the row echelon form of a matrix. The row echelon form is never useful in practice, but it is handy for checking homework in other subjects!!*

5.3 Creating Matrices

Scilab also provides quick ways to create special matrices and vectors.

```
c  =  ones(4,3)
d  =  zeros(20,1)
I  =  eye(5,5)
D  =  diag( [2 1 0 -1 -2] )
L  =  diag( [1 2 3 4], -1 )
U  =  rand(5,5)              // U is a  5 x 5 matrix
                             // of uniformly distributed random numbers.
R  =  rand(5,5,'normal');    // R is a  5 x 5 matrix
                             // of normally distributed random numbers.
```

(More information on each of these commands can be found with `help`.)

Matrix-matrix and matrix-vector multiplication work as expected, provided the dimensions agree. Matrices and vectors can both be multiplied by scalars. In the next example, `B` is another 4×4 matrix and `c` is a column vector of length 4 (i.e. a 4×1 matrix).

```
B = [1 1 0 0
     0 2 1 0
     0 0 3 1
     0 0 0 4 ]        // Rows can be separated by
                      // making a new line as well as using ';'.
c = [1; 0; 0; -1]
5*B                   // Multiply scalars and matrices.
B*c                   // Multiply matrices and vectors.
A*B                   // Matrix by matrix multiplication.
B*A                   // Note: AB is not the same as BA !
```

Some of the following commands do not work. There is no harm in trying them.

```
c*c        // Wrong dimensions for matrix multiplication.
c*A        // Wrong dimensions for matrix multiplication.
c'         // The transpose of c, i.e. a row vector.
c'*A       // Now matrix multiplication is permitted.
c*c'       // This multiplies a  4 x 1  by a  1 x 4  matrix.
c'*c       // What is this quantity usually called?
```

Exercise 10

1. *Calculate* `inv(A)*A` *and* `A*inv(A)`*. Do they give the expected result? (Use the help command if you can't guess what the* `inv` *command does.)*

2. *Verify for the matrices A and B above that $(AB)^{-1} = B^{-1}A^{-1}$.*

3. *Verify that the Scilab command* `A^(-1)` *can also be used to form the inverse of A.*

5.4 Systems of Equations

Consider again the problem of solving the system of equations $Ax = b$. One obvious way that appeals to Mathematicians is to calculate $A^{-1}b$. As in the previous 3x3 case, we should also check that the solution is correct. We do this by calculating the residual $Ax - b$, for our computed solution x. Because of roundoff errors this residual may not be exactly zero, but all its components should be small; say less than 1×10^{-15}.

```
x = A^(-1)*b        // Solve the equation  A x = b
A*x , b             // Check  x  is correct by making
                    //   sure it satisfies the equations.
resid = b - A*x     // Check the residual is zero,
                    //   at least to within roundoff.
```

In fact it is far quicker, and usually more accurate, to solve equations using the backslash operator (\ ) introduced in the first section; rather than calculating with the inverse A^{-1}. The next lines use the backslash command to the solve the equations $Ax = b$, and check that it gives the same answers as those that were calculated using A^{-1}.

```
x1 = A \ b    // This is the fastest way to solve  A x = b
x - x1        // Here both answers are the same.
```

Exercise 11

1. *Solve the system of equations*

$$\begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} x = \begin{bmatrix} 1/4 \\ 1/5 \\ 1/6 \end{bmatrix}$$

using \. *Check that the computed solution does actually satisfy the equations (i.e. check that $b - Ax = 0$, apart from rounding errors).*

2. *Use* `rand(n,m,'normal')` *to generate a 700 x 700 matrix,* `A`*, and a column vector* `b` *of length 700. Solve the system of equations* $Ax = b$ *using the commands*`x = A\b` *and* `x = inv( A ) * b`*, timing each command with the* `timer()` *command. Which command is the faster and why? (Hint: before working with these large matrices use the* `clear` *command to remove all variables from Scilab's memory. This frees up enough memory to squeeze in this largish problem. If you have problems, try a smaller system of equations that fits onto the computer you are using.)*

3. *Set up the* 10×10 *Hilbert matrix. The* ij *component is given by* $1/(i + j - 1)$*. You can use* `feval` *to produce the matrix given the function* `deff('z=f(x,y)','z=1.0/(x+y-1)')`

   Create a right hand side vector b *by* `b = A(:,1)`*. We are now going to solve the equations* $Ax = b$*. This is a common test problem in linear algebra. (What is the true solution* x *to the equations* $Ax = b$*?)*

   (a) *Solve the equations* $Ax = b$ *using the backslash command, and then check that the computed solution satisfies the equations by calculating the residual,* $b - Ax$*.*

   (b) *Now solve the equations by the command* `A^(-1)*b` *and calculate the residual for this second solution.*

   (c) *Which gives the better (i.e. smaller) residual?*

   (d) *Which method gives the smaller error? (Explanation: As* b *is the first column of* A*, you can work out the true value of* x *without computation. (Explain this briefly.) The error is the difference between Scilab's computed solution and the true solution. (The true solution, the one you work out by abstract thought, can be typed directly into Scilab.) The size of the error is the largest absolute value of the components of the error vector.)?*

4. *Extend part (3) by solving a slightly different problem. Let the right hand side vector* b *be the first column of* A*, divided by 3; i.e.* `b = A(:,1)/3`*. (What is the solution to the equation* $Ax = b$ *with this new right hand side?)*

   (a) *What is the size of the error and the size of the residual when the* `\` *is used to solve the system with the new right hand side?*

   (b) *Extend the computation further by letting the right hand side be the* j^{th} *column of* A *divided by 3; i.e.* `b=A(:,j)/3` *for* $j = 1, 2, 3, \ldots, 10$*. Complete the following table.*

Column j	*Size of the Residual*	*Size of the Error*
1	.	.
2	.	.
⋮	⋮	⋮
10	.	.

What conclusions do you draw? If we are worried about the accuracy of a computed solution is it sufficient to check that the solution satisfies the equations to within round off errors? (Hint: If `x_comp` *and* `x_true` *are the computed and true solutions the size of the error can be found from* `max(abs( x_comp - x_true ))`*.*

This exercise should show that the backslash operator \ is both faster and more accurate than the inverse. Clearly it is better to use the backslash and avoid the calculating inverses in numerical work!

6 Polynomials

We can use Scilab procedures to deal with polynomials. The polynomial $a_0 + a_1 x + \ldots + a_{m-1} x^{m-1} + a_m X^m$ can be represented in Scilab symbolically.

In the following example, $-4 - 3x + x^2$ is represented by the vector of coefficients $[-4\ -3\ 1]$ and by a polynomial p We setup the polynomial with the command

```
v = [-4,-3,1]
p = poly(v,'X','coeff')
```

We can also build up a polynomial in a more natural way.

```
x = poly(0,'x')       // Seed a Polynomial using variable 'x'
p = x^2 - 3*x - 4     // Represents x^2 - 3 x -4
```

There are many useful functions that can be applied to polynomials. We can find the roots (numerically) of a polynomial, evaluate a polynomial or find its derivative.

```
z = roots( p )
z(1)^ 2 -3*z(1) -4    //  Two ways to evaluate the poly.
horner(p,z(1))        //    at the first root.
derivat(p,'x')        // Calculate the derivative of the polynomial
```

The Scilab polynomial methods generalise to rational functions. For instance

```
x = poly(0,'x')       // Seed a Polynomial using variable 'x'
p = x^2 - 3*x - 4     // Represents x^2 - 3 x -4
r = x/p               // Represents the rational
                      //    function x/(x^2 - 3 x -4)
```

Exercise 12

1. *The polynomial* $x^2 - 3x - 4$ *has real roots, but* $x^2 - 3x + 4$ *has complex roots. Find these complex roots with* `roots`*.*

2. *Experiment with rational functions, i.e. create, evaluate and differentiate the rational function* $x/(x^2 - 3x - 4)$*.*

7 Graphs

Consider the problem of graphing functions, for example the function

$$f(x) = x|x|/(1 + x^2)$$

over the interval $[-5, 5]$.

7.1 Function Plotting

Scilab provides a simple method for defining functions. For simple functions, the definition can be written at the interactive prompt "on line". For instance to define the function

$$f(x) = x|x|/(1 + x^2)$$

we would write

```
deff('[y]=f(x)',['y= x*abs(x)/(1+x^2)']);
```

This will define a Scilab function with the name `f`. We can then evaluate the function as such

```
f(3)
```

To produce a plot of this function we can use the `fplot2d` command. For instance to plot the function f in the interval $[0, 1]$ we use

```
x = (-5:0.1:5)';
fplot2d(x,f)
```

The first command produces a column vector of x values from 0 to 1 in steps of .1. The second produces the plot of the function `f`.

7.2 Component Arithmetic

The new method is illustrated in the following lines.

```
x = (-5:.1:5)' ;
y = x .* abs(x) ./ ( 1 + x.^2) ;
plot2d( x , y )
```

The first command produces a vector of x values from -5 to 5 in steps of .1. That is the column vector $x = [-5, -4.9, \cdots, 0, \cdots 4.9, 5]'$. The vector y contains the values of f at these x values. As there are so many points the graph of x against y looks like a smooth curve.

The novelties here are the operators `.*` , `./` and `.^` in the second command. These are the so called *component–wise operators.* In the above example `x` is a column vector of length 101 and `abs(x)` is the column vector whose i^{th} entry is $|x_i|$. The formula `x*abs(x)` would be wrong, as this tries to multiply the 1×101 matrix `x` by the 1×101 matrix `abs(x)`. However the component-wise operation `x.*abs(x)` forms another vector of length 101 with entries $x_i|x_i|$. Continuing to evaluate the expression using component-wise arithmetic, we get the vector `y` of length 101 whose entries are $x_i|x_i|/(1 + x_i^2)$. More explicitly

$$\texttt{x} = \begin{bmatrix} -5 \\ -4.9 \\ -4.8 \\ \vdots \end{bmatrix}, \quad \texttt{abs(x)} = \begin{bmatrix} |-5| \\ |-4.9| \\ |-4.8| \\ \vdots \end{bmatrix}, \quad \texttt{x.*abs(x)} = \begin{bmatrix} -5 \times |-5| \\ -4.9 \times |-4.9| \\ -4.8 \times |-4.8| \\ \vdots \end{bmatrix}$$

$$\texttt{x.*abs(x)./(1+x.^2)} = \begin{bmatrix} -5 \times |-5|/(1+(-5)^2) \\ -4.9 \times |-4.9|/(1+(-4.9)^2) \\ -4.8 \times |-4.8|/(1+(-4.8)^2) \\ \vdots \end{bmatrix}.$$

(Of course we should not become overconfident. This rule for dividing by vectors cannot be used in Mathematics proper.)

Note that `p.*q` and `p*q` are *entirely different.* Even if `p` and `q` are matrices of the same size and both products can be legitimately formed, the results will be different.

Exercise 13

1. *(Easy) Find two 2×2 matrices* `p` *and* `q` *such that* `p.*q` $\neq$ `p*q`.
2. *(Hard) Find all 2×2 matrices* `p` *and* `q` *such that* `p.*q=p*q`.

Before more exercises, we show how to add further curves to the graph above. Suppose we want to compare the function we have already drawn, with the functions $x|x|/(5+x^2)$ and $x|x|/(\frac{1}{5}+x^2)$. This is done by the following three additional commands. (Remember `x` already contains the x–values used in the plot. These new commands are most easily entered by editing previous lines.)

```
y2 = x .* abs(x) ./ (5 + x.^2) ;
y3 = x .* abs(x) ./ ( 1/5 + x.^2) ;
plot2d(x,[y y2 y3])
```

Note that `x` needs to be a column vector for this to work.

Exercise 14

1. *Use* SCILAB *to graph the functions* $\cos(x)$, $1/(1+\cos^2(x))$, *and* $1/(3+\cos(1/(1+x^2))$ *on separate graphs.*

2. *Graph* $1/(1+e^{\alpha x})$, *for* $-4 \le x \le 4$ *and* $\alpha = .5, 1, 2$, *on the one plot.*

3. *Use* `plot2d` *to draw a graph that shows*

$$\lim_{x\to 0} \frac{\sin(x)}{x} = 1.$$

   (Hint: Select a suitable range and a set of x values and plot $\sin(x)/x$. *If you get messages about dividing by zero, it means you tried to evaluate* $\sin(x)/x$ *when x is exactly 0.)*

7.3 Printing Graphs

When the graph has been correctly drawn we will usually want to print it. As many graphs come out on the printer, it is a good idea to label the graph with your name. To do this use the `xtitle` command.

```
                                    \\ Be sure to put quotes
xtitle('Exercise 3.4 Kim Lee.')     \\     around the title.
```

Bring the plot window to the front. The title should appear on the graph. If everything is okay, click on the 'file' menu → 'print scilab' item in the graph window. If everything is set up correctly, the graph will appear on the printer attached to your computer.

After printing a graph, it is useful to take a pen and label the axes (if that was not done in SCILAB), and perhaps explain the various points or

curves in the space underneath. Alternatively before printing the graph, use the SCILAB command `xtitle` with two optional arguments and the legend argument of the `plot2d` command for a more professional appearance. For instance

```
xbasc()
plot2d(x,[y y2 y3],leg='function y@function y2@function y3')
xtitle('Plot of three functions','x label','y label')
```

Another option is to use the graphics window file menu item, export, or the `xbasimp` command to print the current graph to a file so that it can be printed later. For example to produce a postscript file use the command

```
xbasimp(0,'mygraph.eps')
```

This will write the current graph (associated with graphics window 0) to the file `mygraph.eps`. This file can be stored on your floppy and printed out later.

7.4 Graphs in Reports

Instead of printing graphs directly, it is sometimes easier to copy Scilab graphs into a word processor such as Windows *Wordpad* or *MS Word*. (A character editor such as *Notepad* is not good enough). Then the graph can be included as a part of a written report.

- The word processor must be open at the same time as we are using SCILAB.
- Move to the SCILAB plot window, and use the FILE→COPY TO CLIPBOARD (ENHMETAFILE) menu subcommand. (This copies the current graph onto the clipboard.)
- Then move to the word processor, place the cursor where graph is to go, and press CTRL-V to paste the graph into the word processor.

Several graphs can be saved in a file this way and incorporated into the final report. Ensure though that the report file is saved on your floppy disk or onto your personal home directory.

If you are working in the computer laboratories, make sure the report file is small enough to fit on a floppy; and make sure you store the report on the floppy and take the floppy when you leave.

7.5 Saving Graphs

The easiest way to save a graph is to save the Scilab command you used to create the graph. These are usually only one or two lines. If the commands are more than one or two lines long, they should probably be saved in a file (see later chapters.)

7.6 Advanced Graphics

The commands given are sufficient to produce clear, simple graphs that suffice for most projects. However, many people are moved to do more:- to alter the size and fonts of labels, to change the thickness of lines etc. If you think this is necessary, more information is available.

- To demonstrate additional graphics features use the help dialog or the demo command

Index