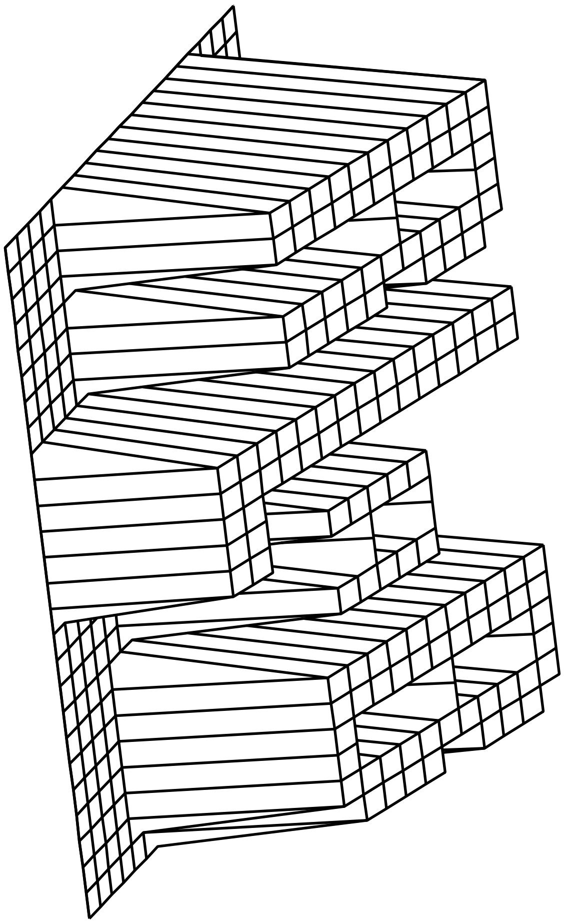


Ian R. Searle &

List of Figures

0.2 [illegible]

an ftp site for R L a B . lo ok in pub/matlab/RLaB.

binary versions for some machines. On the North American continent csi.jpl.nasa.gov acts as

The main ftp site is evans.ee.adfa.oz.au. The directory pub/RLaB contains the sources and

Public License.

and contribute to. o T protect this freedom, copying of the program is protected yb the GNU General

R L a B stands for Our-Lab, since it is intended to b e a freely available program that anyone can use,

0.1 R L a B is freely available a

0 Primer priming

RLaB uses a structured language

discard old programs when confronted by better algorithms that are in use.

minimising the effort required to implement algorithms, it is hoped that you will be more willing to

your solution and hopefully avoid becoming bogged down in low level implementation details. By

Because you enter commands at a high (mathematical) level you can concentrate on figuring out

systems. RLaB allows you to experiment with complex matrix maths in an interactive environment

a form that is freely available and ready to be compiled and used on a variety of common computer

RLaB brings the power of stable matrix maths tools plus a stable data plotting facility together in

examples as you encounter them.

where typewriter dark text is meant to represent the text you would see sitting in front of a

s nlab

A properly installed R R can be started on the terminal

2.1 How to run it

2 Starting to use R

function at which point we hope you will be able to start using R

with quite a few handy functions already built-in, we give examples of their use including the plot be added to the core R

L

B program since you do the source

a

benefit of being able to write your own features to R

before they are executed. This loss efficiency associated with R-file interpretation is traded for the

L

the core of R B. In contrast, R

a

In general, the functions listed in the first group are the most efficient as they are compiled to

R-files that has been setup for you by whoever installed your R

les in the : : :/rlib directory come as a standard part of R

to as "R-files". These R-files are stored in directories, which the help command searches. The help

The subsequent topics refer to commands that has been written in R

```
> help WARRANTY
```

L

In R B . ariables can be names of any length containing most possible characters including

a

2.4 ariable V assignment and displa

complete without a memory so we look at how to store results in a ariable.

where we see that i or j can represent the complex number

```
> 1/1i/1j

> 1/1i * 1/1j

0 - 2i

> 1/1i * 1/1j
```

R B command line, you will find that f " # ! g are the same as f C-b C-p C-n C-f g - think

If your keyboard is missing the arrow keys but C-p did cause previous commands to pop up on the

graphical user interface in one paragraph).

for you, skip the remainder of this section (and count yourself lucky that we weren't describing a

has been created from an old, enter it in the usual way by pressing RETURN. If this has not

the delete key, typing C-d enough to delete the character beneath the cursor. When a new command

commands and then try horizontal cursor motion with the f ! g and try some editing with

previous R B commands are displayed - if they are, then confirm that # also displays more recent

be able to take advantage of command line recall and editing. Try typing the " key to see if you

[illegible] but your keyboard does have the arrow keys f " # ! g,

[illegible] If the word GNU just means any of several African antelopes constituting the genus

modify previous lines - if it works, skip to section 3.

k and press p). If this is successful, test the standard character and word editing commands to as a rather small string (length=1), so the data type boolean (or logical) can be handled by small

true and false are represented by the integers 1 and 0. Just as the data type char can be handled

The observant reader might be wondering what has happened to the boolean data type? In R

```
Hello world
```

```
> hw
```

and the value of variable hw may be printed out by entering

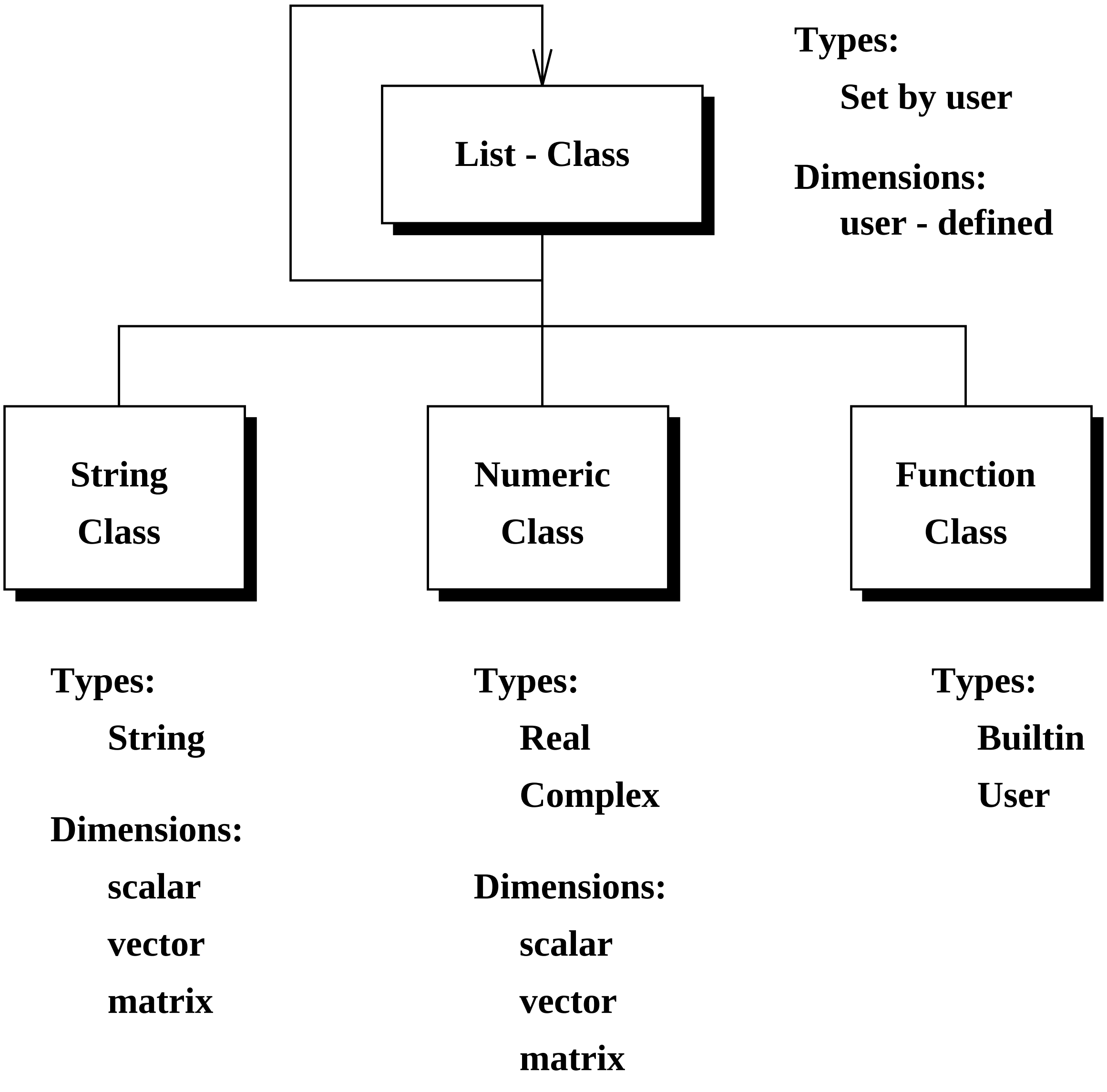
```
Hello world
```

```
> hw = "Hello world"
```

To place a string into a variable, you could enter a statement such as

Just as a number was previously stored in a variable, the same can be done with a string of characters.

Object Hierarchy



3.3 Numerics

The RLaB numeric class includes objects of type real and complex. The numeric object also encompasses objects of scalar, vector, or matrix dimension. If you want to, you can think of all numeric objects as matrices. Thus, a vector is simply a 1xN matrix, and a scalar is a 1x1 matrix. Since the numeric object is most commonly used, it will get the most coverage.

3.3.1 Matrix Creation

The simplest way to create a matrix is to type it in at the command line:

```
> m = [ 1, 2, 3; 4, 5, 6; 7, 8, 9 ]
m =
```

When using vectors, or single dimension arrays, two matrices are created. The simplest way to

arrays.

or program does not need two dimensional arrays, then you can use matrices as singly dimensioned

Although there is no distinct vector type in R

3.3.2 Vector Creation

```
> a = read ( "a.dat" );
```

in one matrix per file, and no variable name information is available. readm is used [illegible]

is most often used to read in data created by other programs. Since readm is only capable of reading

The readm function reads a text file that contains white-space separated columns of numbers. readm make a complex:

ange a matrix attribute, you can do so by changing the data in a. For example: if you want to

In fact it will destroy the dimension of a and create a list with element named nr. If you wish to

Note that these matrix attributes are "read-only". In other words: assignment to a.nr is pointless.

```
real
>          a.type
num
>          a.class
                                    15
>          a.n
                                         5
>          a.nc
```

4 5

> a [2,3 ; 1,2]

T reference a sub-matrix:

9

6

3

> a [; 3]

4 5

> a [2 ;]

T reference an or column:

3.3.6 Matrix Operations

[illegible] [illegible] [illegible] [illegible] the same dimensions.

The [illegible] and column dimensions of the matrices on the RHS, and the matrix description within the

```
7    300    400

4    200    300

1           2

a  =

> a[2,3;2,3] = [200,300;300,400]

7           8

4    200
```

.* performs element-by-element multiplication on its operands. The operands must have the same

element-by-element fashion for each column in the matrix. The result is a NxM matrix.

A or B is a Nx1. and the other is NxM. In this instance the operation is performed

element-by-element fashion for each row in the matrix. The result is a MxN matrix.

A or B is a 1xN. and the other is MxN. In this instance the operation is performed

matrix. The result is a MxN matrix.

A or B is a 1x1. In this case the operation is performed element-by-element on the entire

row and column dimensions. Unless:

- performs element-by-element subtraction on its operands. The operands must have the same

element-by-element fashion for each column in the matrix. The result is a NxM matrix.

A or B is a Nx1. and the other is NxM. In this instance the operation is performed of A. The difference between ' and .' is only apparent when A is a complex matrix; then A'

This unary operator performs the matrix transpose operation. A' swaps the rows and columns

A. For a matrix with complex elements a complex conjugate transpose is performed.

This unary operator performs the matrix transpose operation. A' swaps the rows and columns of

scalar. If both A and B are matrix, then the row and column dimensions must agree.

A.^B raises A to the B power in an element-by-element fashion. Either A or B can be matrix or

performed via A's eigenvalues and eigenvectors. The operation is not valid if B is a matrix.

is performed by successive multiplications. When B is not an integer, then the operation is

A^B raises A to the B power. When A is a matrix and B is an integer scalar, the operation

element-by-element fashion for each column in the matrix. The result is a NxM matrix.

A or B is a Nx1 and the other is NxM. In this instance the operation is performed if you really understand it.

etc. You also read the prop er sections in your text(s), and you have to try your hand at it to see

6

Supp ose you are learning ab out normal-equations, orthogonal transformations, QR decomp ositions,

Now it is time for a few illustrative examples : : :

3.3.8 Examples

this is b ecause any is a vector-oriented function. This will b e discussed later.

all() returns true if all of the elements of its arguments are non-zero. Note that any is used twice;

The function any() returns true if any of the elements of its arguments are non-zero. The function

```
> any ( any (a == 4))
```

try it first-hand. ou Y kno that:

Y ou read ab out the construction of Householder vectors and reflections; you could like to

W this is all to o , easy you can to get , dirt so you move on to orthogonal transformations.

statement I suggest you read the section 'Linear Systems of Equations".

a terrible result with the abevo pro cedure. If you are to pursue the reasoning b ehind the previous

fly the problem we are dealing with is not ill-conditioned , otherwise we may have pro duced

3

2

1

```
x =
> x = inv (a'*a) * (a'*b)
```

```
> P'a
-0.255    -0.018    -0.127    -0.0156   0.958
-0.0952   -0.00671  -0.0474   0.994     -0.0156
-0.776    -0.0547   0.614     -0.0474   -0.127
-0.11     0.992     -0.0547   -0.00671  -0.018
-0.558    -0.11     -0.776    -0.0952   -0.255
```

P =

- P = [illegible]

[illegible]

Now that we have our Householder vector, we are ready to assemble the Householder reflection.

single index when subscripting the variables.

But, this is less than clear. Note that in this case, since we are working with vectors, we only use a enclosed within braces if the expression is true. The expression must evaluate to a scalar-expression.

The if-statement performs a test on the expression in parenthesis, and executes the statements

4.1 If-Statement

r in all of the vector statements.

The vector statements use a syntax that is similar to the C-language

the for-statement, the break-statement and the continue-statement.

vector. The vector statements available in R

We must take a small diversion before proceeding on with the rest of the objects and discuss

4 Program Flow Control

```
2
i =
1
i =
> while ( i < 2 ) { i = i + 1 }
> i = 0;
> while ( 0 ) { "TRUE" }
g
statements
f
while ( [illegible] )
```

usage throughout the remainder of the primer.

Although they will not b e explicitly discussed - there are more examples of [illegible] statemen

```
4
i -
3
i -
1
i -
> for ( i in 1:4 ) { if (i == 2) { continue } i }
3
i -
```

5.1 unction F tlSyn

```
6.72
> list.expsin (2)
expsin logsin
> list.expsin = function ( x ) { return exp (x) * sin (x) }
0.63
> list.logsin (2)
logsin
list = << logsin = logsin >>
```

to group functions that serve a similar purpose, or perform different parts of a larger procedure.

W that represent user-functions can also be part of list objects. Sometimes it can be useful

The different scopes serve to protect data from operations that occur in the other scopes. There is

[illegible] the R L a B session is terminated.

when the function returns. A file's static scope is created when the file is loaded, and remains intact

A function's local scope is temporary: it is created when the function is [illegible] and is [illegible]

environment

There are [illegible] other types of environments available: a function's local environment and a file's static

scope.

[illegible] [illegible] [illegible] added. The workspace will also be referred to as the global-symbol-table or the global

The environment or workspace consists of the built-in functions, and any other variables or functions

When you start a R L a B session, either interactively or in batch mode, you create

5.2 Function Scoping Rules

5.3 Function Argumen Passing

declared global will be bound to the global scope.

if necessary variables declared local will be bound to the function's local scope, and variable

There are two declarations: 'global' and 'local' that can be used to override the default behavior

When 'fun' is created it binds 'A' and 'pi' to file.r's static environment

```
---- end of file.r ----
```

```
fun = function ( a ) { return A*sin(pi*A*a); }
```

```
pi = atan(1)*4;
```

when writing our first function(s) to help us understand how R

describing which variables are local, arguments, global, or file-static. fvscope can be very helpful

RLaB has a special built-in function (fvscope) that analyzes user-functions, and makes a report

function variables as local will prevent accidental destruction of user's global variables.

to declare all variables (other than function arguments or built-in functions) as local. Declaring all

If you wish to write function(s) to serve as often used utilities or libraries, then care should be taken

v so that changes will not affect the caller's variables.

anging any of the function arguments. If necessary the function arguments can be passed by

that the arguments will be modified during function execution, care should be exercised to avoid

arguments types and the function return object. If the function documentation does not clearly state

class and type, if necessary. The documentation (comments) should clearly define the requisite

Statements in a file are executed in the same manner as they would be had they been typed in interactively. Files containing ordinary commands and multiple functions are acceptable. In fact, complete programs can be written and run interactively or in batch mode. To run a program in batch mode you can try:

```
% rlab program.r &
```

Or the program could contain `#!/usr/local/bin/rlab` on the first line. Then, if your operating system provides the proper support, R

background by simply typing:

```
$ chmod +x program.r
$ ./program.r
```

```
return v;
v[1] = 1;
}
    }
        v[2:n] = v[2:n]/b;
    {
    if(u != 0)
    b = v[1] + sign(v[1])*u;
{
if(n > 1)
u = norm(v, "2");
n = max( size(v) );
```

```
-1.49                                   -1.11
> ( p1 = P (house_v (a(:,1)) ) ' a
```

test out our own new functions like so:

This function is a small one, and simply implements the formula we demonstrated earlier. We can

```
    return [ eye(m,m) - 2*(V*V')./(V'*V) ];

    m = max( size(V) );

P = function ( V )

// P: Generate P matrix
```

```
// House.qr returns a list containing 'q', and 'r'

// upper triangular.

// matrices H1,...,Hn, such that if Q = H1*...Hn, then Q'*A = R is

// Given A, with M >= N, the following function finds Householder

// house_qr.r

}

        return A;

        A = A + v*w';

        w = b*A'*v;

        b = -2/(v'*v);
```

```
0.0918          0.441
        0.95          0.915
            0.7          0.96
a    =
>    a
    0.902          0.0735
0.0918          0.441
        0.95          0.915
            0.7          0.96
>    s.q    *    s.r
            q                              r
>    s    =    house_qr    (    a    )
```

6 Objects|Other Data Structures

Although R L a B is primarily a linear-algebra programming tool, other data structures are necessary to allow the user some flexibility and a little extensibility. The two remaining data structures do just that. Strings allow users/programmers to write intelligible error messages, and properly annotate program inputs and outputs, as well as label quantities during program execution. The list object is a very flexible data structure that can hold numeric, string, function and other list objects. Since lists are indexed in an associative fashion, they are a very powerful tool.

6.1 Strings

When performing numeric computations, strings are not normally used for [illegible] except error messages and etc. [illegible] there are many other tasks for which strings are quite useful.

6.2 Lists

W will do a little more with strings in Section 6.2.

```
> strsplit (x)
```

```
> x = "this is a single string";
```

the desired pieces back together again.

allows the user to operate on individual characters in a string; the '+' operator can be used to \glue"

The function strsplit will split an arbitrary string into a matrix of one-character strings. This numbers in the 5th column (the resident process size).

L

function, and R B piping capability to read the output from the UNIX ps command, and sum the

a

separated tokens. The tokens are either numbers or strings. In this example we will use the getline

The getline function reads from a designated file, or pipe, and separates the input into whitespace

6.2.1 Examples

sizes and types, as well as strings, other lists, or user-functions.

return several values. Because of the lists flexible nature a function can return matrices of differing

As [illegible] seen in an earlier example lists can be used within functions when it is necessary to

to string type

The second method allows string and numeric variables to be evaluated before doing the conversion

```
        lcl = 1:mat.nc;
{
if (!exist (cl))
}
        lcl[cl[i]] = i;
}
for (i in 1:cl.n)
}
        if (length (rl) != mat.nr) { error ("mkib: rl wrong size"); }
else
        rl = 1:mat.nr;
{
```

```
return mlb.m[ir; icl]
}
    }
        icl[i] = mlb.cl.[cl[i]];
    }
    for (i in 1:cl.n)
else
    icl = 1:mlb.m.nc; // use all the columns
}
if (!exist (cl))
```

matrix-functions. The list below gives examples from each type of function, but is not all-inclusive.

Many of the defined functions fall into one of three categories: scalar-functions, vector-functions, or

7.1 Function Files

filename is the name of the R-file to load, without the .r extension.

The files displayed by the rfile command can be loaded by typing rfile filename', where

will list all of the R-files in the directories listed in the environment variable RLAB_SEARCH_PATH.

'what()', this will list all of the currently available functions. Typing 'rfile' from within R

load and compile the function or statements. To see what functions are currently available type:

encountered. Instead, the user must explicitly load, or use the rfile command to get R

RLaB does not automatically search RLAB_SEARCH_PATH for R-files when an unresolved reference is

portable libraries using the built-in functions.

There are owt yp es f I/O in R

8 Input and Output

format write read readm

what tic to c pth

system getline sfso who

ywhere else. Some examples are:

Mac. unctionf These functions are in this particular category simply b ecause they don't t

```
> for (i in 5:1:-1) { fprintf (output, "%i\n", i); } close (output);
1
2
3
4
5
```

If the fprintf output were not sent to the Unix program sort, we would expect the output to appear in descending order. In fact, fprintf does write the output in descending order, to sort. Then, sort re-orders its input, and writes it to stdout. The same trick works in reverse; functions that expect input can get it from a sub-process. A good example is getline:

```
> input = "| ls -la *.[ch]";
> fsum = 0;
```

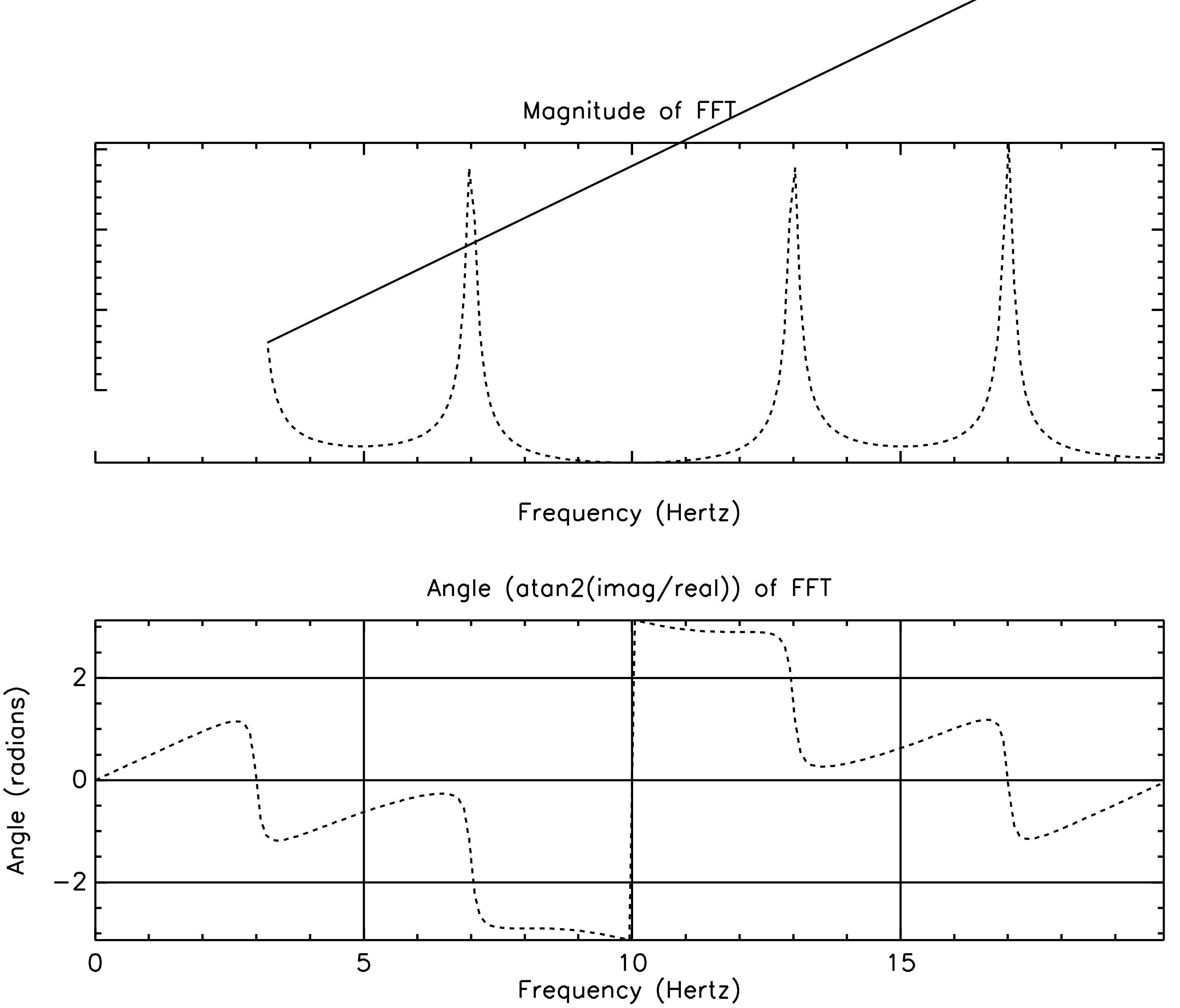
9.1 2-D Plotting

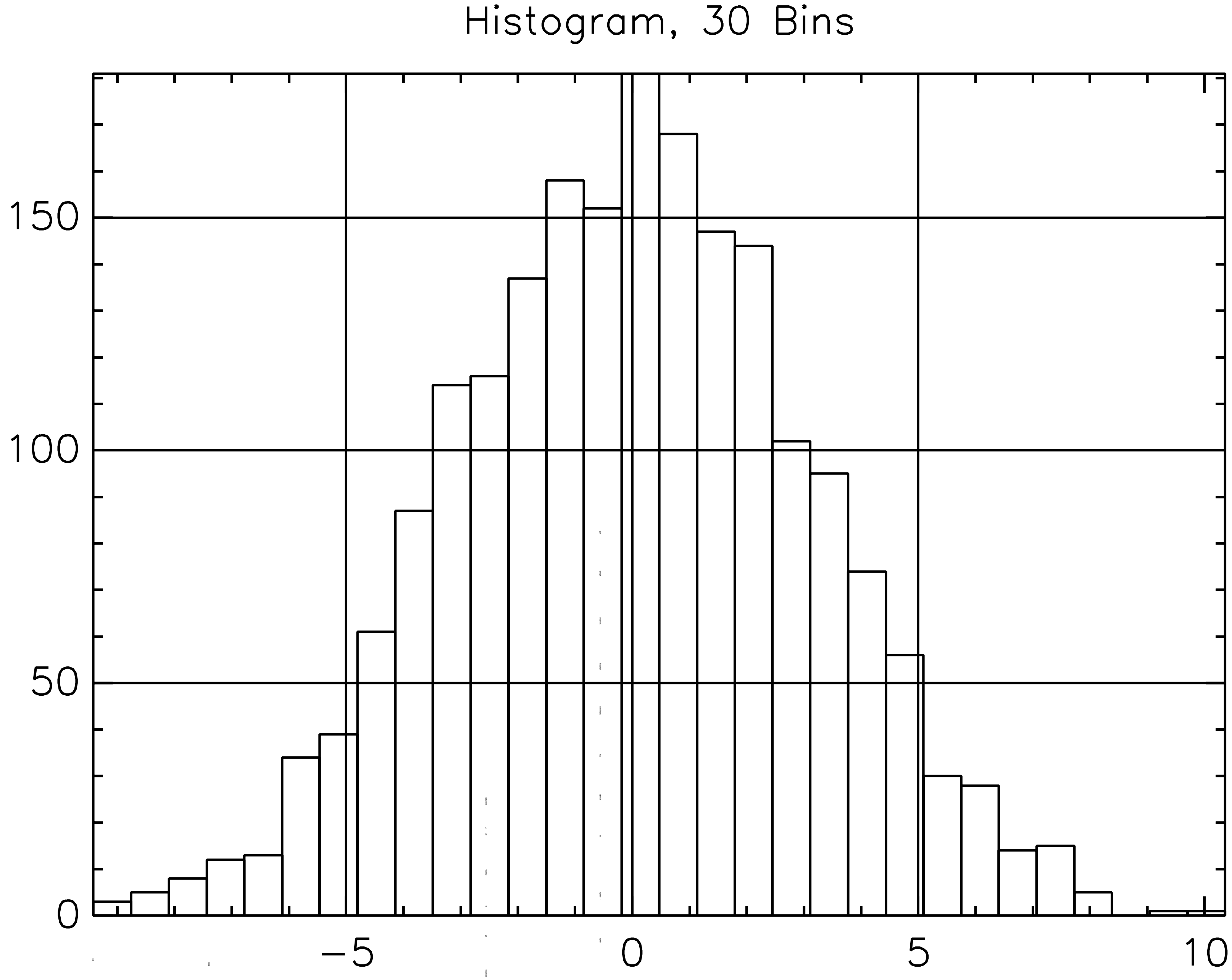
To plot some data with a minimum of effort do:

```
> data = (0:pi:.1)';
> data[;2] = cos(2*pi*data*3);
> pstart (,,"xwin");
> plot (data);
```

The pstart function is called with the default sub-plot definition (1 subplot), and the X-window as the display device. The plot function accepts matrices, or lists as arguments and plots the columns of a matrix, or the columns of each matrix of a list.

Now for a slightly more complex example.

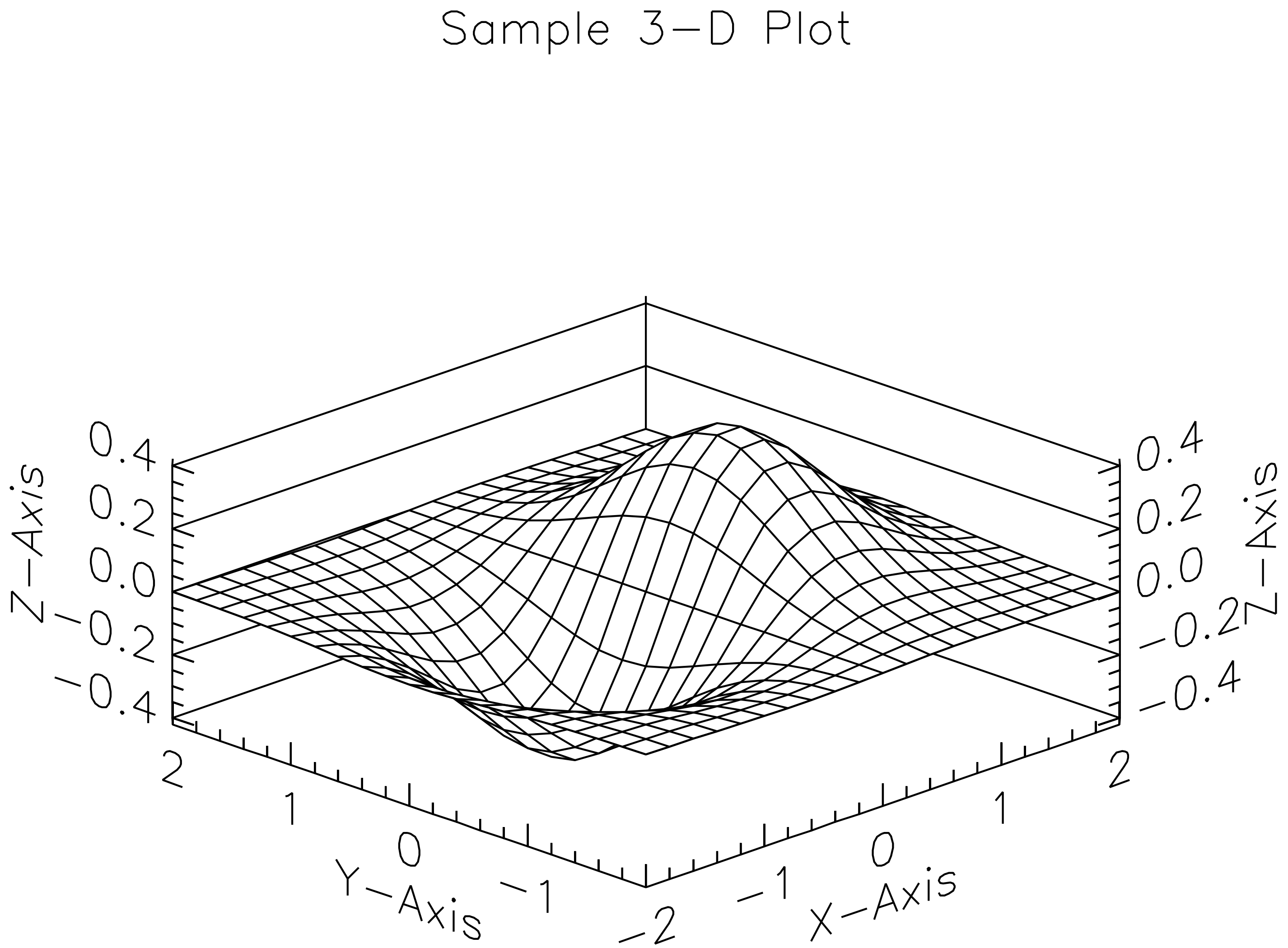

Magnitude of FFT
Frequency (Hertz)
Angle (atan2(imag/real)) of FFT
Angle (radians)
2
0
−2
0
5
10
15
Frequency (Hertz)


Histogram, 30 Bins
150
100
50
0
−5
0
5
10

9.3 3-D Plotting

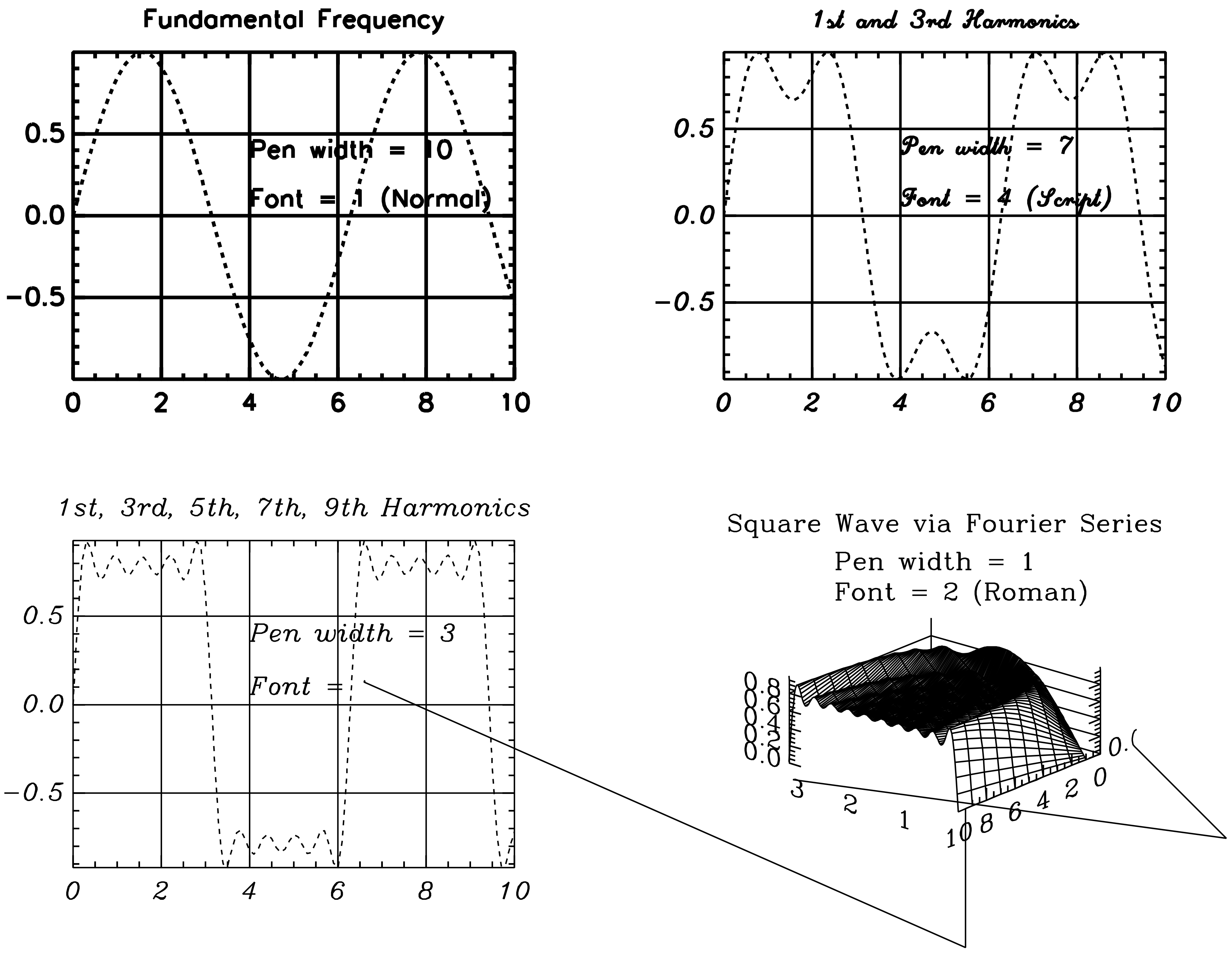
Plotting 3-dimensional data is quite straightforward, and very similar to the steps used for plotting 2-dimensional data. We will begin with an example:

```
> // Create the data
> X = -2:.2:2;
> Y = X;
> Z = [];
> for (i in 1:X.n) {
> for (j in 1:Y.n) {
> Z[i;j] = X[i] * exp (-X[i]^2 - Y[j]^2);
> }
> }
```


Sample 3-D Plot
Z-Axis
0.4
0.2
0.0
-0.2
-0.4
Z-Axis
0.4
0.2
0.0
-0.2
-0.4
Y-Axis
2
1
0
-1
-2
X-Axis
-1
0
1
2

```
plptex ("Pen width = 10", 4, 0.4);

plot ( [t,y] );

plfont(1);

plwid(10);

pltitle ("Fundamental Frequency");

y = sin (t);

t = (0:10:.1)';

// Compute the 1st term in the Fourier series and plot.

plstart(2,2, "xwin");

// Start up a plot window...
```

```
% of adding more and more terms to the series.
% Now make a nice 3-D plot that shows the effect

    y((k+1)/2,:) = x;
    x = x + sin(k*t)/k;

for (k in 1:19:2)

x = zeros(size(t));
y = zeros(10,max(size(t)));
```



Fundamental Frequency

1st and 3rd Harmonics

1st, 3rd, 5th, 7th, 9th Harmonics

Square Wave via Fourier Series

10 Summary

RLaB can do quite a few things that we do not cover here. The reference manual or the online help should be consulted for more detailed descriptions of language and function behavior. We will close out this primer with a listing of the currently available functions (both built-in and user- functions).

```
> what()
abs        epsilon    log
acos       error      log10
asinh      eval       logspace   plprint
all        [illegible]           b
any        exp        [illegible]
```