Chapter 7: User Manual

7.1 Introduction

This chapter is a guide to the Command Line Interface of the software. There are two versions of the software (based respectively around the System Manager 1 and System Manager 2 frameworks). I will begin by discussing the use of version 1 (V1), as all that has to be said on this is equally relevant to version 2 (V2). This covers the use of the Calculator Manager. The version 2 discussion will extend the coverage to the System Manager, the Data Manager and the Graph Manager. The System Manager is not visible to the user in version 1.

7.2 Version 1 of the evaluation and visualisation tool

The startup screen shows the commands available for use in the Data Manager. This is to be skipped by typing the following at the Data Manager prompt:

```
Data Manager> return
```

The Graph Manager interface is then activated, which must also be skipped using:

```
Graph Manager> return
```

These Managers are invoked initially to allow testing, and although fully operational are not used during the calculation and graphing of equations. See Version 2 for this functionality.

7.2.1 V1 Startup

V1 is now ready to go, and the following command screen for the Calculator Manager is displayed:

```
Commands Available:
     add <name>
     remove <name>
     set_current <name>
     storage
     errors
     clear_errors
     clear_memory
     clear <name>
     resetmanager
     verify
     auto_verify_on
     auto_verify_off
     dump
     return
     order <expression> | <assignment> | <equation>
     help


Calculator Manager.NULL>
```

At startup there are no calculators in existence, and so the .NULL name is displayed in the prompt. The system is not very useful in this state, as most commands are disabled. The command:

```
Calculator Manager.NULL> help
Calculator Manager.NULL> help
```

displays the commands list for the Calculator Manager.

7.2.2 Creating, destroying and viewing calculators in the Calculator list

In order to do any calculations, a new calculator must be created by the user:

```
Calculator Manager.NULL> add mycalc
```

The calculator name must begin with an alphabetic character, and may then consist of any combination of letters and numerals, of unlimited length. This naming convention is adopted across the whole application. A name must be unique to the type of object being created. For example, all calculator names must be different, but they do not have to be different to names of data sets, set inputs, etc. (See V2).

The prompt now changes to:

```
Calculator Manager.mycalc>
```

This shows that the Calculator Manager has accepted the ‘add’ command and has created a new calculator named ‘mycalc’. Any calculator commands are now directed to this. Further calculators may be added in this way, and each time a new one is added, the prompt is updated to show the name of the new calculator, thus switching the command target. For example:

```
Calculator Manager.mycalc> add yourcalc
Calculator Manager.yourcalc>
```

This creates a new calculator, and shifts the target to it. The calculator referenced in the prompt is called the ‘current calculator’ - as this is the calculator currently targetted for accepting commands.

To change the target calculator back to ‘mycalc’, the following command is used:

```
Calculator Manager.yourcalc> set_current mycalc
Calculator Manager.mycalc>
```

To display all calculators currently available, use:

```
Calculator Manager.mycalc> dump

Calculator 'mycalc' contains:
Variable List:
 This u-list is empty
EQUATION List:
 This u-list is empty

Calculator 'yourcalc' contains:
Variable List:
 This u-list is empty
EQUATION List:
 This u-list is empty
```

All calculators are displayed in alphabetical order, along with their personal variable and equation lists. At present these contain no items.

To display the contents of the current calculator, use:

```
Calculator Manager.mycalc> storage
Calculator 'mycalc' contains:
Variable List:
 This u-list is empty
EQUATION List:
 This u-list is empty
```

To delete a calculator, the remove command is used with the calculator's name:

```
Calculator Manager.mycalc> remove mycalc
Calculator Manager.yourcalc>
```

In this case, the target calculator is removed, and the target is moved to the calculator with the lexically smallest name. If there are no other calculators available, the .NULL prompt is redisplayed.

If an invalid name is used in any of the above commands, the command is refused and the one of the following error messages is displayed: (where 'e' was the invalid name)

```
Manager ERROR - e calculator was not removed because 'e'
calculator doesn't exist
Manager ERROR - setting current calculator failed as calculator
with name 'e' does not exist
```

Similar errors are displayed where the user issues a 'dump' or 'storage' command when no calculators exist.

Calculators may be added and removed at will by the user.

7.2.3 Evaluating Expressions

To use the current calculator to evaluate an expression:

```
Calculator Manager.yourcalc> order 10*j+pi-e
result is: '4.233108e-01+1.000000e+01j'
```

This expression contains references to the three constants stored in all calculators. Their values are always available, and cannot be changed. The complex number result is displayed in standard form, using 6 digits precision. The output form of a result may not be changed by the user. The expression may contain white space between terms. The expression is read up to the newline character.

Expressions are parsed using standard precedence rules. See next section.

The constants stored in each calculator are:

Constant name	Complex number
e	2.7182818284590452354 + 0j
pi	3.1415926535897932385 + 0j
j	0 + 1j

7.2.4 Operators and Functions permitted in an expression:

The following operators and functions are recognised by the calculator:

Operators:

+	Addition				
-	Subtraction and Unary Minus				
*	Multiplication				
/	Complex Division				
^	Complex Power (infix operator eg 2^3=8)				
rootx	Xth Complex Root (infix operator eg X xroot 3 calculates the Xth root of 3.				
logx	Logarithm to base X (infic operator eg X xlog 3 calculates base X log of 3				
!	Integer Factorial, postfix				
MILLI	Engineering Symbol, multiplies prefixed term by 10^-3				
MICRO	“	“	“	“	by 10^-6
NANO	“	“	“	“	by 10^-9
PICO	“	“	“	“	by 10^-12
FEMTO	“	“	“	“	by 10^-15
KILO	“	“	“	“	by 10^3
MEGA	“	“	“	“	by 10^6
GIGA	“	“	“	“	by 10^9
TERA	“	“	“	“	by 10^12
PETA	“	“	“	“	by 10^15
EXA	“	“	“	“	by 10^18

()	Parentheses: nesting allowed up to any level.
[]	Complex Modulus brackets, calculates magnitude of expression contained within. Unlimited nesting is permitted, but as with (), they must be matched.

Standard algebraic precedence is assumed. Rootx, logx and power operations are of equal precedence, and higher than multiplication and division. Engineering symbols have highest precedence, except for parentheses and modulus.

Functions:

sqrt cbrt	Complex square root and cube root of postfixed operand.
sin, asin	Sine and inverse sine of postfixed operand
cos, acos	Cosine and inverse Cosine of postfixed operand
tan, atan	Tangent and inverse Tangent of postfixed operand
sinh, asinh	Hyperbolic sine and inverse sine of postfixed operand
cosh, acosh	Hyperbolic cosine and inverse cosine of postfixed operand
tanh, atanh	Hyberbolic tangent and inverse tangent of postfixed operand
ln	Natural logarithm of postfixed operand
log10	Log to base 10 of postfixed operand
log2	Log to base 2 of postfixed operand
arg	Complex argument of postfixed operand
re	Returns real component of postfixed operand
im	Returns imaginary component of postfixed operand

WIND (variable name, lower bound, upper bound)
This function evaluates to 1+0j if the named variable lies between the bounds. Otherwise it evaluates to 0+0j.

Trigonometric and logarithmic calculations may not be performed on complex numbers containing non-zero imaginary components.

Power and all root operations upon complex numbers return the primary root.

Compound Functions:
Only one instance of one of these functions may be present in a single expression.

SUM (variable name, upper bound) expression
SUM (variable name, lower bound, upper bound) expression

These two functions allow summation using a single control variable, ranged between the integers lying between 0 (or a specified lower bound) and an upper bound. The expression is evaluated repeatedly for all integer values within the bounds, and the sum of these evaluations is returned.

PROD (variable name, lower bound, upper bound) expression
PROD (variable name, upper bound) expression

These two functions allow summation using a single control variable, ranged between the integers lying between 0 (or a specified lower bound) and an upper bound. The expression is evaluated repeatedly for all integer values within the bounds, and the product of these evaluations is returned.

References:
Names referencing constants, variables and equations can be used as operands in any of the above. If the reference is invalid the expression is not evaluated and 0+0j is returned.

Numbers:
Numbers may be used in fixed point and scientific forms (eg 1.30e4). Evaluation precision is limited to the size of a 'double' in the C++ compiler used for compiling the program. Numbers of up to modulus 1e120 may be used with the CodeWarrior compiler.

If a function is given invalid operands, eg tan pi/2, divide by 0, or 0^0 then the error is reported and 0+0j result returned.

7.2.5 Assigning Variables

A variable may be assigned in the following way:

```
Calculator Manager.yourcalc> order <name>=<some expression>
eg
Calculator Manager.yourcalc> order x=10
Calculator Manager.yourcalc> order x=10+3*y
```

The name must not be used already in the equation or constant lists. If already in the variable list, the variable is assigned the evalution of the expression. If not already in the variable list, the variable is added to the list of variables belonging to the current calculator, along with the evaluation of the associated expression. If the expression contains errors, the variable is assigned 0+0j. Note that the expression is evaluated immediately, and if any names references contained within do not exist, 0+0j will be assigned.

7.2.5 Defining equations

An equation may be assigned in the following way:

```
Calculator Manager.yourcalc> order <name>:<some expression>

eg
Calculator Manager.yourcalc> order z:10+3*y
```

The name must not be used already in the variable or constant lists. If already in the equation list, the expression associated with the name is updated to the new expression. If not already in the equation list, the equation name is stored with the expression (as a string of characters) in the equation list.

Equations are not evaluated immediately and so can refer to variables and equations which are not yet defined. If they remain undefined when the equation is evaluated (by having its name referenced in an expression) an error is reported and the value 0+0j returned.

An equation must not contain a circular reference. An equation must not reference itself, and must not reference equations that directly or indirectly refer back to itself.

It is impossible to evaluate a set of equations that contain a circular reference.

7.2.6 Equation Verification

To automatically check all equations in the target calculator for circular references whenever a new equation is defined, use the following:

```
Calculator Manager.yourcalc> auto_verify_on
```

If active, the Calculator will refuse to store any equation that completes a circular reference.

To deactivate this feature use:

```
Calculator Manager.yourcalc> auto_verify_off
```

This is for testing only. When auto-verification is off, a verification of all equations in the target calculators equation list may be performed using:

```
Calculator Manager.yourcalc> verify
```

If an auto or manual verification fails, an error trace is displayed in the form:

```
“a b c a”
```

In this case, the validator has found that equation ‘a’ references equation ‘b’, which references equation ‘c’, which contains a reference back to ‘a’ producing a circular reference.

7.2.7 Error reporting

Each calculator has an error report which buffers all errors flagged by the calculator. These may arise due to invalid name references, invalid operands or invalid syntax. The errors are reported in plain English and reflect the nature of the error.

The error report is limited to a 1000 character size. If this limit is reached, further error messages are not recorded.

The error report for the target calculator may be viewed by:

```
Calculator Manager.yourcalc> errors
```

and my be cleared for the target calculator allowing for the storage of more error messages by:

```
Calculator Manager.yourcalc> clear_errors
```

A count is kept of the number of errors that have occurred since the error report was last cleared. If errors occur during evaluation, the total count is displayed on the screen after evaluation has terminated.

7.2.8 Clearing variables and equations

Individual variables and equations may be removed from the target calculator using:

```
Calculator Manager.yourcalc> clear <name>
```

Where ‘name’ is to be cleared.

All variables and equations may be removed from the target calculator using:

```
Calculator Manager.yourcalc> clear_memory
```

7.2.9 Re-initialising and leaving the Calculator Manager

To clear all calculators from the Calculator Manager use:

```
Calculator Manager.yourcalc> resetmanager
```

To exit the Calculator Manager use:

```
Calculator Manager.yourcalc> return
```

Note that all calculators and their equation and variable lists are maintained after exiting the Calculator Manager.

7.2.10 Drawing graphs of equations stored in Calculators

In order to draw a graph, a calculator called ‘graph’ is required in the Calculator Manager. This calculator must contain the following variables, set according to how the user wants the graph to be scaled, and the data used to evaluate the co-ordinates of the graph. Note that only the real components of the variables are processed:

Variable	Purpose
xmin	Minimum visible x-axis value
xmax	Maximum visible x-axis value
xscale	Major Divison scale, difference in x-axis value between scale numbers.
xdiv	Number of minor divisions between each major division. (small tick mark)
ymin	Minimum visible y-axis value
ymax	Maximum visible y-axis value
yscale	Major Divison scale, difference in y-axis value between scale numbers
ydiv	Number of minor divisions between each major y-axis division. (small tick)

A graph’s points are calculated by the System Manager setting a variable called ‘var’ in the current target calculator set in the Calculator Manager to a range of values, and for each evaluation the value of two equations stored in the target calculator, xgraph and ygraph, are extracted. The real components of the results are used as x-y co-ordinates for each point.

So, the following variables are also required in the ‘graph’ calculator:

Variable	Purpose
varmin	Smallest input data value
varmax	Largest input data value
res	The resolution determining how many data values are processed in the varmin to varmax range.

In addition the following equations are required and must have some reference, either direct or indirect, to the ‘var’ variable, in order to draw a meaningful graph:

Equation	Purpose
xgraph	Equation providing x co-ordinate for the current ‘var’ value.
ygraph	Equation providing y co-ordinate for the current ‘var’ value.

Exiting the Calculator Manager invokes the System Manager to draw a graph based upon the above data. The current target calculator in the Calculator Manager is used to store ‘var’ variable values and extract ‘xgraph’ and ‘ygraph’ equation results.

Once the graph is drawn, the System Manager returns the user to the Calculator Manager, and the process is repeated. All calculators and associated variables and equations are preserved since the last exit from Calculator Manager.

7.3 Version 2 of the evaluation and visualisation tool

7.3.1 V2 Start-up

The following is displayed at start-up and may be re-displayed using the ‘help’ command:

```
Commands Available:
     graphman
     dataman
     calcman
     graphprocess
     setprocess
     viewprocess
     doprocess
     reset
     help
     quit

System Manager>
```

The ‘calcman’ command invokes the Calculator Manager in exactly the same way as Version 1 of the system, except once the user exits from the Manager, control is passed back to this screen and no graphs are drawn. The notion of a ‘graph’ calculator containing graph parameters is to be disregarded in Version 2 of the software.

The ‘dataman’ and ‘graphman’ commands invoke the Data Manager and Graph Manager interfaces respectively. Exiting from these interfaces returns the user to this screen. Remaining commands concern the evaluation and graphing of data and are left until the end.

7.3.2 Data Manager

The following screen is displayed upon entry:

```
Commands Available:
     newrecord <record_name> <field_name>...
     delrecord <record_name>
     records
     newdataset <data_name> <record_name> <length>
     deldataset <data_name>
     datas
     newinput <input_name> lower_index upper_index start_value step
     delinput <input_name>
     inputs
     newmap <map_name> <input_field_name>... (newline) <output_field_name>...
     delmap <map_name>
     maps
     resetmanager
     help

Data Manager>
```

Lists of records, data sets, set inputs and IO maps are maintained.

The user may display all records, datas, inputs and maps by giving the appropriate command:

```
Data Manager> records
Data Manager> datas
Data Manager> inputs
Data Manager> maps
```

The user may remove by name and record, data set, set input or IO map by using one of the following commands followed by the name of the object:

```
delrecord,  deldataset, delinput, delmap.
```

The user may add a new record describing a set of column names by :

```
newrecord <record_name> <field names>
```

Where record_name is unique to the records already stored in the Data Manager, and fields names is a space separated list of names terminated by a newline.

The user may add a new data set holding a data array of user-defined length, with columns labelled using a particular named record:

```
newdataset <data_name> <record_name> <length>
```

Where data_name is unique to the data sets already stored in the Data Manager, record_name references a record currently stored in the Data Manager and length is the integer length of the array.

The user may add a new set input holding a description of how to load a data set column with real numbers:

```
newinput <input_name> lower_index upper_index start_value step
```

Where input_name is unique to the set inputs already stored in the Data Manager, lower_index and upper_index indicate the row bounds of the input data, start_value is the real number to store in the lower_index, and step is the difference between value stored in two adjacent rows.

The user may add a new input output map using:

```
newmap <map_name> <input_columns> newline
<output_columns>
```

Where map_name is unique to the IO maps already stored in the Data Manager, input columns are the names of columns to be read from a data set and given to a calculator, whilst output columns are the names of columns referencing equation evaluations to extract from the calculator. Note that there is a newline separating the space separated list of input columns and the space separated list of output columns.

The user may clear all records, data sets, set inputs and IO maps by using: `resetmanager`

The user may exit the Data Manager by using: `return`

All objects defined in the Data Manager are retained upon exit.

7.3.3 Graph Manager

The following screen is displayed upon entry:

```
Commands Available:
     newspec <spec_name>
     delspec <spec_name>
     specs
     resetmanager
     return
     help


Graph Manager>
```

A graph specification is added to the Graph Manager using the following format:

```
Graph Manager> newspec graph1
Horizontal:
Scale Division? 2
Inter-Divisions? 1
Range Min? 0
Range Max? 10
Vertical:
Scale Division? 5
Inter-Divisions? 4
Range Min? 0
Range Max? 20
```

The spec name is unique to the Graph Manager. The interface interactively requests the graph spec parameters from the user.

A graph spec is deleted from the Graph Manger by using the `delspec` command.
A list of all graph specs stored in the Graph Manager may be obtained using the `specs` command. The specs list may be cleared of all specs by using the `resetmanager` command.
The user may exit the graph Manager by using: `return`.

All specs are retained upon exit.

7.3.4 System Manager 2

Returning to the System Manager it is now possible to discover how the whole system works.

```
Commands Available:
     graphman
     dataman
     calcman
     graphprocess
     setprocess
     viewprocess
     doprocess
     reset
     help
     quit

System Manager>
```

The ‘process’ holds a specification for evaluating a set of equations using a set of input variables, storing the results in a data set, and graphing these results. To do an evaluation, the process parameters must first be set up using:

```
setprocess
```

Invoking this command outputs the current process parameter settings and the following prompt:

```
Change which field?
    (c)aluator_name <name>,
    (d)ata_set_name <name>,              (g)raph_spec_name <name>,
    (s)et_input_names <name1> <name2>..., (m)ap_names <name1> <name2>...
```

One of the following is then inputted:

`c <name>`
: set the name of the calculator in Calculator Manager used for evaluation

`d <name>`
: set the name of the data set in Data Manager to store input and output data

`s <name1> <name2> ... newline <columnname1> <columnname2> ...`
: Apply set inputs with name1, name2, etc to columns of data set labelled columnname1, columnname2, etc. Set input 1 is applied to columnname1, and so on. Number of set input names must equal number of column names.

`m <inputname1> <inputname2> ... newline <outputname1> <outputname2> ...`
: Select the columns in the data set to give to the calculator before evaluation. The column names reflect the names of the variables set in the calculator before each evaluation. Select the columns in the data set used to store evaluation results from the calculator. The column names reflect the names of equations used to evaluate outputs.

`g <name>`
: set the name of the graph spec in Graph Manager used to set up the graph axes.

The setprocess command has to be invoked five times to set all parameters. This allows individual parameters to be changed easily.

All objects referenced by name in the process parameters must be present in the respective Managers. All objects are cross-referenced to make sure they are compatible - for example it is not possible to apply a set input that loads 100 elements to a data set only 10 elements long

All column names referenced must be valid for the named data set. They must also represent variables and equations in the calculator used for evaluation.

Parameters aren't checked until the `doprocess` command is invoked. This command requires no parameters. The data set is filled with input data and the calculator is used to evaluate results based upon this data, which are stored back into the data set. For a detailed explanation of how this is achieved please see the System Manager 2 High Level Design.

The `graphprocess` command requires a graph type, '`rect`' or '`cornu`':

```
System Manager> graphprocess rect
<x-axis> <y-axis> <start_index> <end_index>
```

Given the rect parameter, the user is requested for the names of the columns in the data set to plot, across a range of rows in the data set. An x-y graph is plotted using the data.

```
System Manager> graphprocess cornu
<axis-variable> <start_index> <end_index>
```

Given the cornu parameter, the user is requested for the name of one column of data to plot, with a range of rows. The data is split into its real and imaginary components and used to draw a vector line.

The user must store origin information in the Major Divisions field of the graph spec and scale information in the Number of Minor Divisions field of the graph spec for both horizontal and vertical axes. Scale determines how many pixels per unit vector component are drawn, and origin selects the point (in terms of unit vector components) where the cornu spiral plot is to be started plotting from. A vector plot of the complex data is then drawn.

Once the graph is drawn, the user remains in the System Manager. If equations are to be refined, the Calculator Manager is entered and the appropriate calculator has its equations changed. Returning back to the System Manager and repeating the doprocess command will do a repeat calculation, but this time using the refined equations.

Once a data set has been loaded from data using a `doprocess` command, differently scaled graphs can be obtained without re-evaluating results by changing the graph spec referenced in the process. An invokation of `graphprocess` then re-plots the data using the new axes.

If different input data is required, the user enters the Data Manager, sets up a new set input object, and references this new set input back in the process of the System Manager. Due to the system being able to keep track of multiple objects (especially data sets), many complex systems can be refined in memory at once.