DYALOG

The tool of thought for expert programming

Dyalog APL for Raspberry Pi User Guide

Dyalog APL Version 14.0

Dyalog Limited

Minchens Court, Minchens Lane
Bramley, Hampshire
RG26 5BH
United Kingdom

tel: +44(0)1256 830030

fax: +44 (0)1256 830031
email: support@dyalog.com
http://www.dyalog.com

Version 14.0

Revision: 20140314_140

Contents

1 About This Document

This document is intended for anyone who wants to run Dyalog APL on a Raspberry Pi.

This document does not provide information on the Raspberry Pi or Dyalog APL – information and documentation for these products can be downloaded from http://www.raspberrypi.org/ and http://docs.dyalog.com/ respectively.

The information provided in this document supplements that given in the *Dyalog APL for UNIX Installation and Configuration Guide* and the *Dyalog APL for UNIX User Guide* (available to download from http://docs.dyalog.com). In cases where the information given is different between the two documents, this document should be regarded as the definitive source for Dyalog APL on the Raspberry Pi.

1.1 Audience

It is assumed that the reader has a basic understanding of Linux on the Raspberry Pi. No prior knowledge of Dyalog APL is required.

For those who are new to Dyalog APL, some guidance on resources that can help with getting started is included in Chapter *5*.

2 Introduction

Dyalog APL – the tool of thought for expert programming – is often used in research and business applications where it is important to integrate domain experts in the software development process. One of the reasons why Dyalog APL is selected is that applications written using Dyalog APL often out-perform and outscale competing products written using compiled languages, even though compiled code should theoretically out-perform interpreted code. Dyalog APL, the only APL interpreter for the Raspberry Pi, allows the benefits of APL to be leveraged on the Raspberry Pi by bringing the concise, expressive power and performance of Dyalog APL to the compact, credit-card sized Raspberry Pi computer.

With the appropriate sensors and attachments, Dyalog APL can turn the Raspberry Pi into much more than a computer, for example, a temperature monitor, a proximity indicator or a robot. The progress of a robot being developed at Dyalog can be monitored through the following blogs:

- Dyalog developer blog: http://www.jasonrivers.co.uk/category/dyalog/ includes information on how to replicate the robot's hardware
- Dyalog CTO blog: http://www.dyalog.com/blog/category/robots

Some examples of code written for the Raspberry Pi using Dyalog APL are included in Appendix *C*.

The licence for Dyalog APL on the Raspberry Pi is for non-commercial use only. To license Dyalog APL on the Raspberry Pi for commercial use, please contact sales@dyalog.com.

3 Getting Started

This chapter covers the system requirements and installation information for installing 32-bit Unicode Dyalog APL on the Raspberry Pi, as well as the information needed to upgrade or remove an installation.

Only the 32-bit Unicode edition of Dyalog APL is available for the Raspberry Pi.

Dyalog APL on the Raspberry Pi can be accessed from PuTTY or other terminal emulators (as described in the *Dyalog APL for UNIX Installation and Configuration Guide*) or from another Linux desktop. It is not possible to use RDP, VNC or any X-window emulators from Microsoft Windows.

3.1 Pre-requisites

Dyalog APL can be installed and run on the following Raspberry Pi versions:

- Model A (256 MB RAM)
- Model B (256 MB RAM)
- Model B Rev. 2 (512 MB RAM)

This document assumes that one of these Raspberry Pi versions is available and that it has at least 100 MB available in the root file system for the installation of Dyalog APL – another 100 MB is also required temporarily for the installation image.

The instructions in this document assume that the Raspberry Pi is running a Debian (or Debian-based, for example, Raspbian) operating system.

3.2 Installing Dyalog APL on the Raspberry Pi

The information in this section is also available at http://packages.dyalog.com/.

To install Dyalog APL on the Raspberry Pi:

1. Open a terminal window and log in as the root user – change to the root user with the following command:
```
$ sudo su
```

2. Configure a repository using the following commands:
```
$ wget -O - http://packages.dyalog.com/dyalog-apt-
                          key.gpg.key | apt-key add -
$ echo 'deb http://packages.dyalog.com wheezy
       main' > /etc/apt/sources.list.d/dyalog.list
```

3. Install Dyalog APL using the following commands:
```
$ apt-get update
$ apt-get install dyalog-unicode
```

4. Read and accept the licence agreement. For non-commercial use, accepting the licence agreement entitles you to a free, unregistered version of Dyalog APL for Raspberry Pi. To license Dyalog APL on the Raspberry Pi for commercial use, please contact sales@dyalog.com.

Dyalog APL can now be run on the Raspberry Pi (see Section *4.1*).

The following command can be used to see all the available packages:

```
$ apt-cache search dyalog
```

3.3 Upgrading Dyalog APL on the Raspberry Pi

Dyalog APL updates are included with system updates and can be performed at the same time.

To upgrade to a later release or version of Dyalog APL:

1. Open a terminal window and log in as the root user – change to the root user with the following command:
   ```
   $ sudo su
   ```
2. Determine the packages available to upgrade using the following command:
   ```
   $ apt-get update
   ```
3. Upgrade the identified packages using the following command:
   ```
   $ apt-get upgrade
   ```

Dyalog APL is now upgraded.

3.4 Uninstalling Dyalog APL from the Raspberry Pi

If necessary, Dyalog APL can be uninstalled from the Raspberry Pi.

To uninstall Dyalog APL from the Raspberry Pi:

1. Open a terminal window and log in as the root user – change to the root user with the following command:
   ```
   $ sudo su
   ```
2. Uninstall Dyalog APL using the following command:
   ```
   $ apt-get purge dyalog-unicode
   ```

Dyalog APL is now uninstalled.

Uninstalling Dyalog APL does not destroy the repository configured in Section *3.2*.

3.5 Having Multiple Versions of Dyalog APL on the Raspberry Pi

Multiple versions of Dyalog APL can be installed on the Raspberry Pi. To install additional versions, change to the root user and enter the following command:

```
$ apt-get install dyalog-unicode-<version-specific number>
```

where `<version-specific number>` is a three-digit number indicating the version. Possible values are:

- `132` (for version 13.2)
- `140` (for version 14.0)

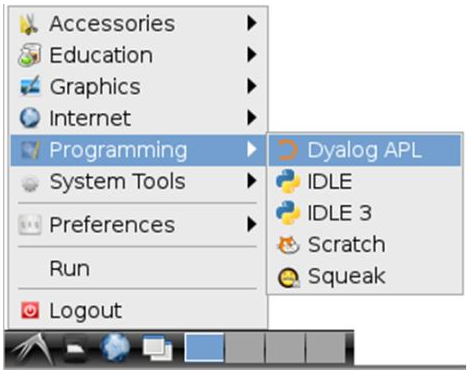
4 Dyalog APL on the Raspberry Pi

This chapter covers the commands necessary to initialise and use 32-bit Unicode Dyalog APL on the Raspberry Pi.

4.1 Initialising Dyalog APL on the Raspberry Pi

Dyalog APL can be started on the Raspberry Pi in either of the following ways:

- In a terminal window, enter `dyalog`
- From the start menu, select **Programming > Dyalog APL**



Both of these methods display a new session window in which Dyalog APL can be run. The session window is identical to the standard Linux non-GUI Dyalog APL session window as documented in the *Dyalog APL for UNIX User Guide*.

If multiple versions of Dyalog APL are installed on the Raspberry Pi (see Section *3.5*), then entering `dyalog` in a terminal window will start the latest installed version. Alternative versions can be started by entering **/opt/mdyalog/<version>/32/unicode/mapl**.

4.2 Configuring Dyalog APL on the Raspberry Pi

Some configuration is a matter of choice (for example, the colour scheme). However, failure to configure the font set can result in Dyalog APL not displaying correctly.

By default, the Migration Level □ML (that is, the degree of migration of Dyalog APL towards IBM's APL2) is 1 (this is reflected in this document and code samples that are available on the open source repository https://github.com/APLPi).

Although Dyalog intends to use a migration level of 1 in all future documentation and tool development, some of the distributed code and earlier documents have not yet been adjusted and could assume other levels.

4.2.1 Configuring the Font Set

Dyalog APL uses Unicode APL glyphs which do not display well in all fonts. Dyalog recommends using the APL385 Unicode font which is included with the installation.

To enable the session window to display the complete range of glyphs, this font must be used – it is installed as part of the Dyalog APL installation and must be set as the default terminal font the first time that Dyalog APL is run on the Raspberry Pi.

Setting the default terminal font to be the APL385 Unicode font impacts all terminal windows. However, as it is a monospace font the effect of doing this is minimal.

To configure the default terminal font to be the Dyalog APL font:

1. In the Dyalog APL session window, select **Edit > Preferences**
   The **Preferences** window is displayed.

2. In the **Style** tab of the **Preferences** window, click on the **Terminal Font** option. By default, this is `Monospace`.

   The **Font Selection** window is displayed.

3. From the **Family** list, select **APL385 Unicode**.

4. Click **OK** to confirm your selection and be returned to the **Preferences** window.

5. Click **OK** to be returned to the session window.

6. In the Dyalog APL session window, press the **Control+L** keys simultaneously to refresh the window and restore the text that was previously visible.

4.2.2 Changing the Background Colour

This section is not specific to Dyalog APL on the Raspberry Pi and contains instructions that can be performed at any time. Changing the background colour of the default terminal window also changes the background colour of all terminal windows.

To change the background colour of the terminal window:

1. In the Dyalog APL session window, select **Edit > Preferences**
   The **Preferences** window is displayed.

2. In the **Style** tab of the **Preferences** window, click on the **Background** option. By default, this is black.

   The **Pick a Colour** dialog box is displayed.

3. Select the colour to use as the background colour by doing one of the following:

   - left-clicking the appropriate colour in the triangle on the left-hand side
   - specifying the Red/Green/Blue values on the right-hand side.

4. Click **OK** to confirm your selection and be returned to the **Preferences** window.

5. Click **OK** to be returned to the session window.

6. In the Dyalog APL session window, press the **Control+L** keys simultaneously to refresh the window and restore the text that was previously visible.

4.2.3 Changing the Colours for Syntax Colouring

During input, the Dyalog APL session window employs different colours for different syntactical purposes. For example, an entry between APL quotes is a different colour to an entry that cannot be evaluated.

Before changing the font colours:

1. Open a terminal window

2. Copy **/opt/mdyalog/<version>/32/unicode/apltrans/xterm** to a suitable location, for example, **/home/<name>/dyalog/apltrans/**:
   $ mkdir –p ~/dyalog/apltrans
   $ cp /opt/mdyalog/14.0/32/unicode/apltrans/xterm ~/dyalog/apltrans/

3. Open the **~/.config/dyalog/config** file and add the following line:
   `export APLTRANS=/home/<name>/dyalog/apltrans:$APLTRANS`

   The **dyalog** directory is automatically created under the **XDG_CONFIG_HOME** directory (by default this is **~/.config**) the first time Dyalog APL is run.

4. Save the amended **config** file.

To change the font colours:

1. Open the **~/dyalog/apltrans/xterm** file for editing.

2. Locate the appropriate entry and change it as required (see Section *4.2.3.1*).

3. Save the amended **xterm** file.

4.2.3.1 Colour Values

The colour definitions in the **xterm** file use precisely-defined formats that follow the convention *<what is being defined> : <definition> <colour> m*. This can be appended with + and a comment if required.

For font colours, the definitions are formatted as: *27 91 51 56 59 53 59 <colour> 109*

To determine the <colour> value:

1. Use http://en.wikipedia.org/wiki/File:Xterm_256color_chart.svg to identify the required colour's number (1-255).

2. Take each digit in the number individually and add 48 to it.

   The resulting set of one, two or three double-digit numbers is the <colour> value. For example, 162 has a <colour> value of 49 54 50 and 82 has a <colour> value of 56 50.

4.3 Using Dyalog APL on the Raspberry Pi

All the information that is needed to run Dyalog APL on the Raspberry Pi is available from the Dyalog APL session window.

When you open the Dyalog APL session window, the `intro` workspace is displayed. The first time that you do this, introductory information is provided. This information can be displayed again at any time using the following command:

```
)load intro
```

Four other commands can be used to display different categories of information:

`get_started`	useful commands to get you started (see Section *4.3.1*)
`APLKeys`	APL keyboard layout (see Section *4.3.2*)
`workspaces`	example workspaces to try (see Section *4.3.3*)
`references`	useful websites (see Section *4.3.4*)

These commands can only be run from within the `intro` workspace.

4.3.1 get_started

The `get_started` command displays a list of valid commands that can be used to manipulate workspaces. These include:

`)load <myws>`	loads `<myws>` into the Dyalog APL session
`)save <myws>`	saves active workspace as `<myws>`
`)save`	saves active workspace
`)clear`	clears active workspace
`)reset`	resets state indicator
`)ed item`	opens the Dyalog APL editor in the session window for the specified item/items
`)off`	exits the Dyalog APL session window

4.3.2 APLKeys

The `APLKeys` command displays an image of the keyboard showing the keys used for Dyalog APL glyphs. This mapping applies to all Linux keyboard layouts.

Users who are familiar with the Dyalog APL user interface under Microsoft Windows should familiarise themselves with any differences between their usual keyboard layout and the Linux keyboard layout.

The image of Linux keyboard mappings is reproduced in Appendix *B*. This image is also available in the **xterm** file in the **/opt/mdyalog/<version>/32/unicode/aplkeys** directory.

To enter APL glyphs in a workspace, hold down the Windows key while pressing the appropriate character key. When there are two glyphs on a key, hold down the Windows key

while pressing the appropriate character key to get the lower of the two and hold down the Windows key and the Shift key while pressing the appropriate character key to get the upper of the two.

4.3.3 workspaces

The `workspaces` command displays some example workspaces that can be loaded using the `)load <workspace>` command. These include:

`sudoku`	sudoku solver for 2-D and 3-D puzzles
`life`	animated Conway's Game of Life
`echo`	TCP echo test (runs on port 7000)

4.3.4 references

The `references` command displays a list of websites that can assist with understanding and learning Dyalog APL. Further useful references are located in Appendix *A*.

4.4 Logging the Session

By default, a session logfile called **default.dlf** is created in the current working directory. This binary file's default name and location can be overridden.

To override the session logfile name and location:

1. Open a terminal window.
2. Open the **~/.config/dyalog /config** file and add the following line:
   `export LOG_FILE=/<path>/<logfilename>.dlf`
3. Save the amended **config** file.

5 Starting with Dyalog APL

If you've not used Dyalog APL before then welcome! Dyalog APL is a concise and powerful language that has built a following of passionate advocates and it's always good to welcome someone new into the community.

Dyalog APL needs to be understood for its elegance and simplicity to be appreciated – in this way it is like every other coding language. Dyalog recommends the following two resources, used together, to start your APL journey:

- Try Dyalog APL online: http://www.tryapl.org/

  TryAPL is a free application in which many of the functions and operators that form the core of Dyalog APL can be tested.

  When accessing the TryAPL application using Midori, the information in the left-hand side of each tab needs to be highlighted to be displayed. This rendering issue is due to the browser not fully supporting CSS/Javascript.

- "Mastering Dyalog APL": http://www.dyalog.com/mastering-dyalog-apl.htm

  "Mastering Dyalog APL" is a complete guide to Dyalog APL. Its clear explanations and tutorials help the reader to advance from novice to specialist. Sample exercises are provided throughout – many of these can be performed in the TryAPL user interface. Alternatively, try them using the Dyalog APL session window on your Raspberry Pi. "Mastering Dyalog APL" is available as a free pdf download.

Additional resources that can help you as you progress further with Dyalog APL are detailed in Appendix *A*. Some examples of what can be achieved using Dyalog APL on the Raspberry Pi are included in Appendix *C*.

Appendix A Useful Resources

The following links provide information that might be useful when mastering Dyalog APL and enhancing the capabilities of your Dyalog APL-driven Raspberry Pi:

- Dyalog APL documentation: http://docs.dyalog.com/
- Dyalog APL sample dfn workspaces: http://dfns.dyalog.com/
- Try Dyalog APL online: http://www.tryapl.org/
- Dyalog APL forums: http://forums.dyalog.com/
- "Mastering Dyalog APL": http://www.dyalog.com/mastering-dyalog-apl.htm
- Open source repository for code samples: https://github.com/APLPi
- Raspberry Pi add-ons: http://quick2wire.com/
- British APL Association journal: http://vector.org.uk
- APL Wiki pages: http://aplwiki.com/
- Editor keycodes and their common keystrokes are included in the appendices of the *Dyalog APL for UNIX User Guide*. This information is also included in the **xterm** file located in the **/opt/mdyalog/<version>/32/unicode/aplkeys** directory.

Dyalog APL is also available for Linux (x86/x86_64), Microsoft Windows and IBM AIX. For more information, see http://www.dyalog.com/

Appendix B Linux Keyboard Mappings

Figure *1* shows the Linux keyboard key mappings used for Dyalog APL glyphs.

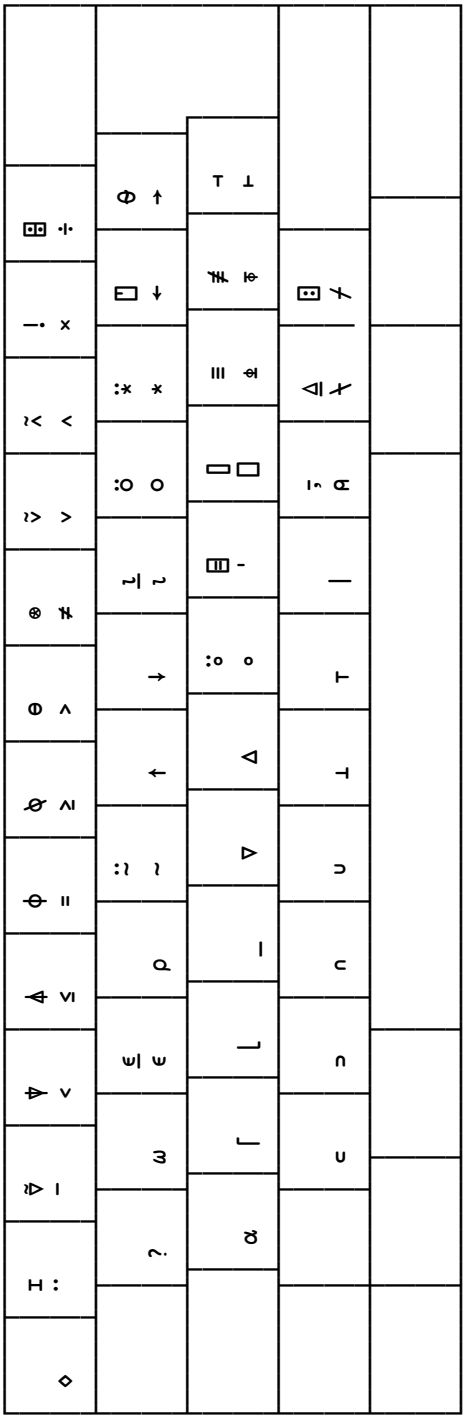


Figure 1. **A Linux keyboard showing the Dyalog APL glyph mappings**

Appendix C Example Code

This appendix includes example code to illustrate some uses of Dyalog APL on the Raspberry Pi.

Sample code for Dyalog APL on the Raspberry Pi is available from the open source repository https://github.com/APLPi. This repository includes the code described in this appendix as well as further examples, and will be maintained and extended as the Dyalog APL for Raspberry Pi project develops.

C.1 Morse Code

This example code enables the Raspberry Pi to utilise a blinking LED to communicate messages in Morse code. It utilises quick2wire software and hardware; instructions for adding quick2wire as a Raspbian software source are available at http://quick2wire.com/articles/how-to-add-quick2wire-as-a-raspbian-software-source.

For the example code to produce the required output, the quick2wire gpio-admin tool needs to be installed. If quick2wire has been added as a Raspbian software source, then this can be done with the following command:

```
sudo apt-get install quick2wire-gpio-admin
```

The quick2wire interface board used in this example is detailed at http://store.quick2wire.com/raspberrypi-interface-board.html.

To run the example code:

1. Download the Dyalog APL code file **Morse.dyalog** , the text file **MorseCode.txt** and the scripted namespace file **Files.dyalog** from https://github.com/APLPi/GPIO onto your Raspberry Pi. Assuming that you have cloned this Github repository to your home directory, these three files will be located in the **/home/<username>/** directory.

2. Open a Dyalog APL session window and load the example code using SALT, for example:
   ```
   ]load /home/<username>/Morse
   ```

3. Initialise the Morse code table. For example:
   ```
   Morse.Init '/home/<username>/MorseCode.txt'
   ```

4. Call the function GPIO with the string to be output, that is,
   ```
   Morse.Display '<string>'
   ```

   For example, to output the Morse code characters SOS, enter `Morse.Display 'SOS'`. An example of the output from this SOS command is available to view at http://youtu.be/xKBS0gtSDQ8.

The **MorseCode.txt** file contains the Dyalog APL code for converting letters, numbers and basic grammatical characters into Morse code.

The Dyalog APL code in **morse.dyalog** is as follows:

```
:Namespace Morse

    ⍝∇:require =/Files

    rate←200÷1000 ⍝ length of "dit" = 200ms ("dah" will be 3x
                                                          this)
    gpio_pin←18   ⍝ Use GPIO pin 18

    ∇ Init filename;t
      ⍝ Read MorseCode.txt file (actual name passed as arg)
      t←⎕FMT #.Files.GetText filename ⍝ Read file into matrix
      Chars←t[;1],' '      ⍝ Chars are in col 1 (and add space)
      Codes←(↓0 1↓t)~¨' '  ⍝ Codes are everything from column 1
      Codes,←⊂'  '         ⍝ Space is as a double long pause
    ∇

    ∇ Display message;direction;value;folder;m;didah;is;
                                        index;output;duration
      ⍝ Output Morse code using LED connected to GPIO pin

      is←{(⍕⍵)⎕NREPLACE ⍺ 0}      ⍝ Write to offset 0 of a file
      index←Chars⍳message         ⍝ Look message up in Chars

      :If ∨/m←index>⍴Chars        ⍝ Any chars not found?
          ('UNSUPPORTED CHARS: ',m/message)⎕SIGNAL 11
      :Else ⋄ output←∊Codes[index],¨',' ⍝ one string, w/ inter-
                                                    char pauses
      :EndIf

     folder←'/sys/devices/virtual/gpio/gpio',⍕gpio_pin
     ⎕SH'gpio-admin export ',⍕gpio_pin ⍝ Creates "direction"
                                                    and "value"
     direction←(folder,'/direction')⎕NTIE 0 ⍝ Open the GPIO
     value←(folder,'/value')⎕NTIE 0         ⍝             files
     direction is'out'                   ⍝ We are doing output

     :For didah :In output
       duration←(1 3 3 7)['.-, '⍳didah]
        value is didah∊'.-'       ⍝ Turn light on if dit or dah
        ⎕DL duration×rate         ⍝ Wait
        value is 0                ⍝ Turn off
        ⎕DL rate                  ⍝ Inter-didah pause
      :EndFor

      ⎕NUNTIE direction value
      ⎕SH'gpio-admin unexport ',⍕gpio_pin
    ∇

:EndNamespace
```

C.2 Average Dice Throw Values

This example code enables the Raspberry Pi to plot its output using SharpPlot (SharpPlot is a graphics package that is included with the Dyalog APL installation – it supports the production of high quality charts in a variety of output formats using a simple set of Dyalog APL functions). No additional hardware is required to generate output from this example code.

To run the example code:

1. Open a Dyalog APL session window and enter the following code:

```
⍝ Load SharpPlot (needed for graphical output)
 )load sharpplot

⍝ Define function for cumulative average of throws
CumAvgRolls←{(+\?⍵⍴6)÷⍳⍵}
```

2. Optionally, test your function:

```
CumAvgRolls 10
```

This should generate output similar to the following ten values:

```
1 3 3 3.25 3 2.666666667 3 3.25 3.555555556 3.5
```

3. Label the graph axes and set the line type

```
⍝  Instantiate a SharpPlot object
sp←#.new #.SharpPlot

⍝ Add a heading to the plot - (⎕UCS 10) is line feed
sp.SetHeading 'Cumulative Avgs of',(⎕UCS 10),'100 Dice
Throws'

⍝ Add a label to the Y-axis
sp.SetYCaption 'Average'

⍝ Add a label to the X-axis
sp.SetXCaption 'Throws'

⍝ Use only solid lines
sp.SetLineStyles #.LineStyle.Solid
```

4. Generate and plot your values using SharpPlot:

```
⍝ Roll 3 dice 100 times each and plot results
sp.DrawLineGraph ⊂CumAvgRolls¨3⍴100

⍝ Save SVG file
sp.SaveSvg '<path>/<filename>.svg'
```

A file called **<filename>.svg** is saved in the specified **<path>**. Opening this file displays a graph similar to that shown in Figure *2*.

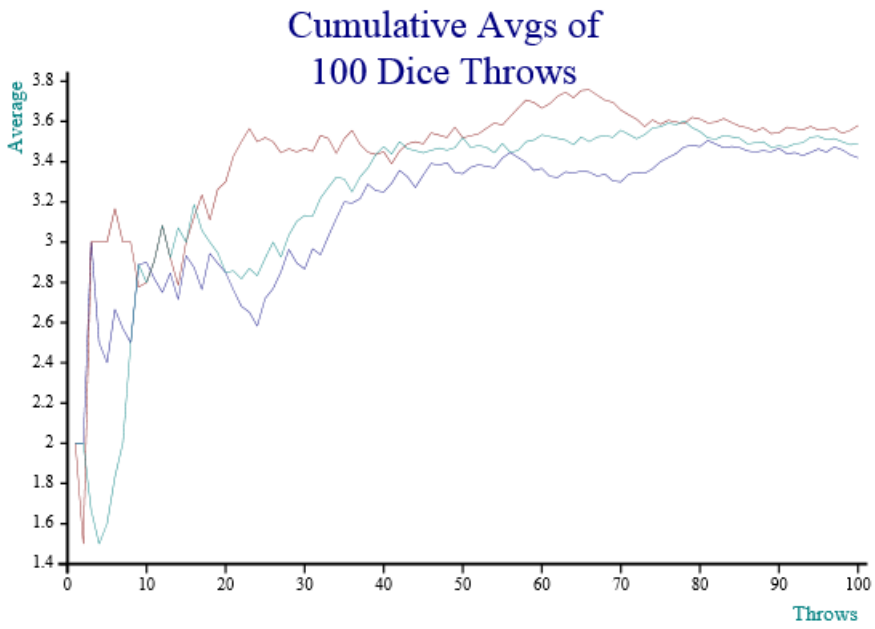


***Figure 2*. Graphical representation of cumulative averages**