VIRTUAL COMPUTING

Da Vinci

For Technical Computing

Users Manual

Shaheen Hoque
11/25/2022

Developed by Shaheen Hoque, Da Vinci is a Graphical Application that provides an integrated environment where Hyper scripts can be edited and run. It also provides tools to process data, produce plots, and format texts. In addition, it includes a file browser.

Contents

Table of Contents

1 Introduction

Leonardo Da Vinci (LDV) software can be used to perform technical computations such as analysis, simulation, post-processing, and visualization. It consists of a graphical Integrated Development Environment (IDE), an object-oriented scripting language named Hyper, numerical function libraries for math, and 2D and 3D plotting tools. LDV is written in Java.

The scripting language, Hyper, is a general-purpose object-oriented interpreted scripting language with an emphasis on technical computation. Hyper combines powerful object-oriented programming (OOP) with an easy-to-use interpretive environment. Hyper can be used in two different modes: interactive and batch script. The interactive mode has some shell-like commands (similar to Linux/Unix/DOS) which provide utility functions. In addition to the scripting, programs can also be written in Java, compiled, and imported into the Hyper's interpretive environment. The imported classes in the compiled java binary can be accessed as easily as accessing the classes written in Hyper scripting language. In fact, any pre-compiled java classes can be accessed this way, including all the public classes in the Java API. Hyper is a full-fledged object-oriented language. It also features strong typing with default parameter values and multiple parameters return from a function call. Hyper supports a rich set of data types related to mathematics and other technical computation, such as Matrix, Vector, Polynomial and several others, in addition to the data types found in a typical programming language. Hyper features a large library of functions, such as Linear Algebra and Statistics, in addition to the basic math functions. Hyper is documented in the Hyper Reference Manual.

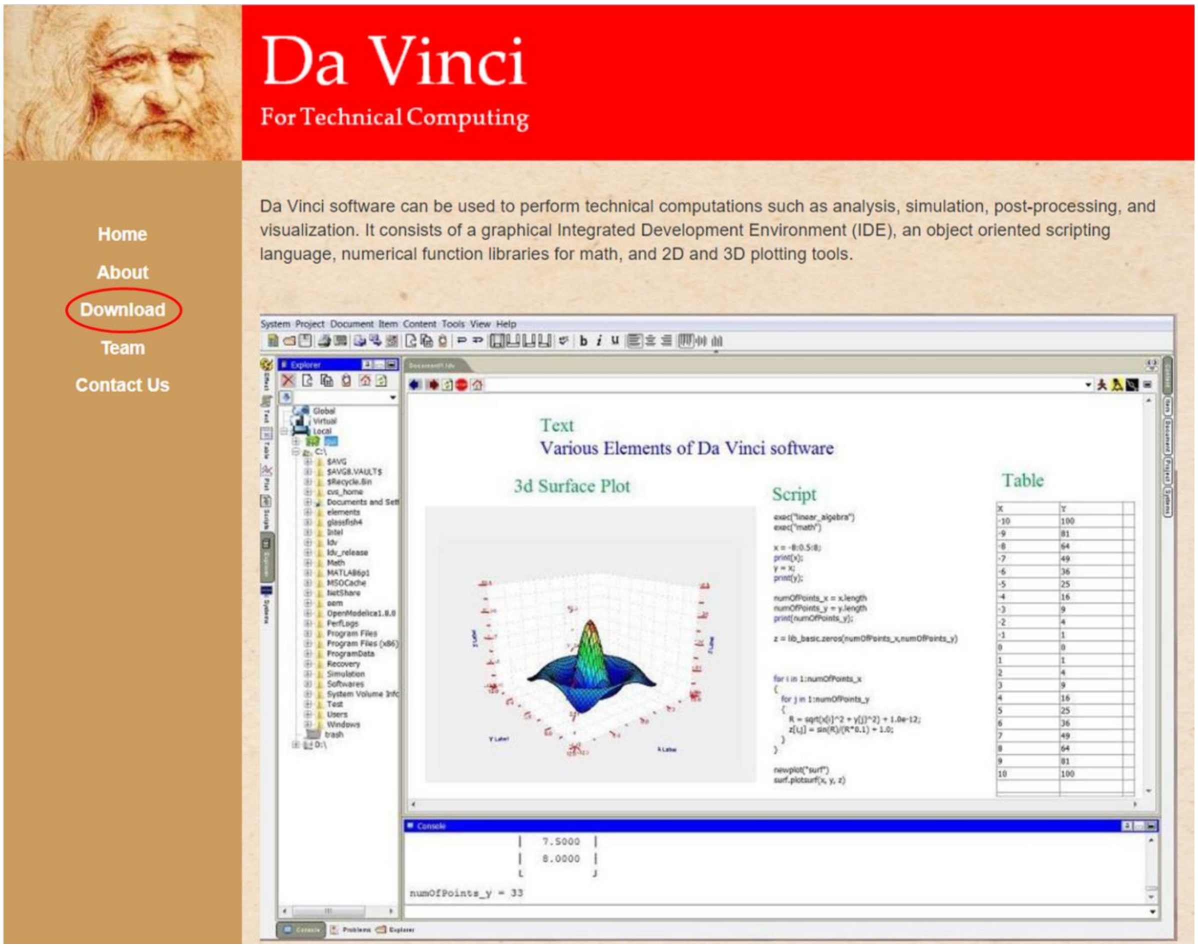
2 Installation

Installation of LDV involves downloading the software from the website and unzipping it into a folder on a local drive.

2.1 Downloading

1. Go to www.virtual-computing.net

2. Click the link labeled “Download”



Da Vinci

For Technical Computing

Da Vinci software can be used to perform technical computations such as analysis, simulation, post-processing, and visualization. It consists of a graphical Integrated Development Environment (IDE), an object oriented scripting language, numerical function libraries for math, and 2D and 3D plotting tools.

Text

Various Elements of Da Vinci software

3d Surface Plot

Script

```
exec("linear_algebra")
exec("math")

x = -8:0.5:8;
print(x);
y = x;
print(y);

numOfPoints_x = x.length
numOfPoints_y = y.length
print(numOfPoints_y);

z = lib_basic.zeros(numOfPoints_x,numOfPoints_y)


for i in 1:numOfPoints_x
{
  for j in 1:numOfPoints_y
  {
    R = sqrt(x[i]^2 + y[j]^2) + 1.0e-12;
    z[i,j] = sin(R)/(R*0.1) + 1.0;
  }
}

newplot("surf")
surf.plotsurf(x, y, z)
```

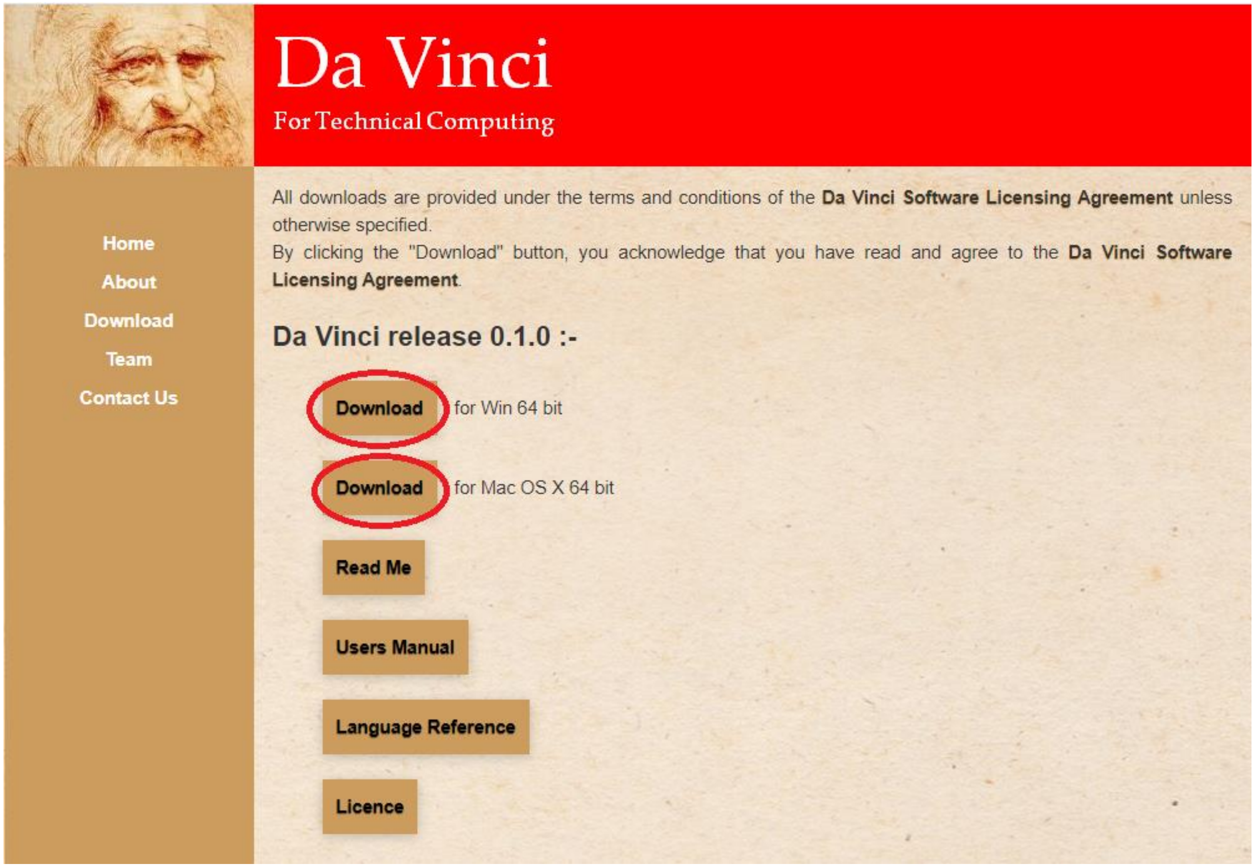
Table

X	Y
-10	100
-9	81
-8	64
-7	49
-6	36
-5	25
-4	16
-3	9
-2	4
-1	1
0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

```
      |   7.5000  |
      |   8.0000  |
      L           J

numOfPoints_y = 33
```

3. Click on the Download button for PC or Mac

4. The downloaded file is named “ldv_release.zip” for PC and “ldv_mac.zip” for Mac.



All downloads are provided under the terms and conditions of the **Da Vinci Software Licensing Agreement** unless otherwise specified.
By clicking the "Download" button, you acknowledge that you have read and agree to the **Da Vinci Software Licensing Agreement**.

Da Vinci release 0.1.0 :-

Download for Win 64 bit

Download for Mac OS X 64 bit

2.2 Installation

Unzip the ldv_release.zip file into your folder of choice on a local drive.

Warning: The software must be installed on a local drive. The software will not perform if installed on a network drive.

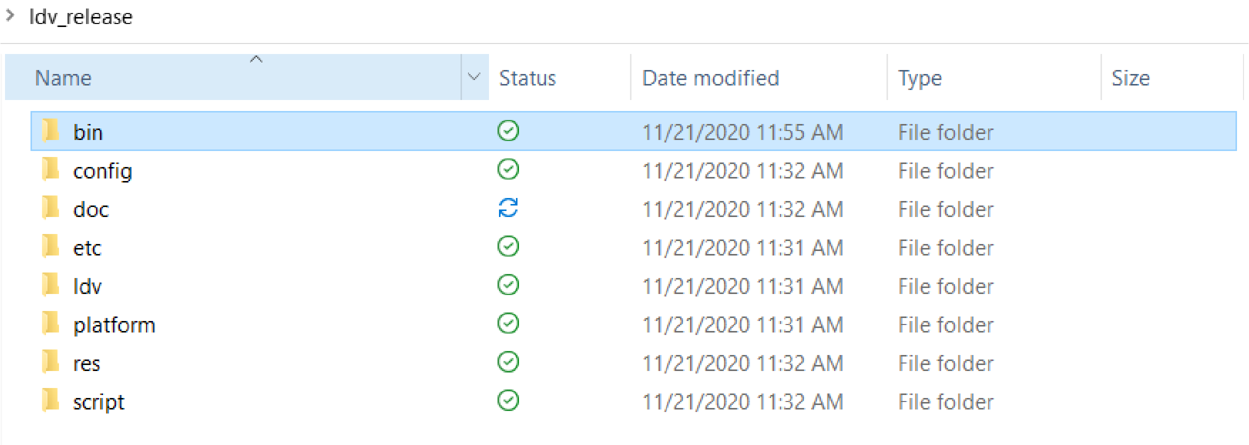
2.3 Directories

2.3.1 Application for PC

This folder contains LDV-based programs written by users in Java. An example simulation file is included by default.

2.3.2 ldv

This folder contains binary files for the software itself.

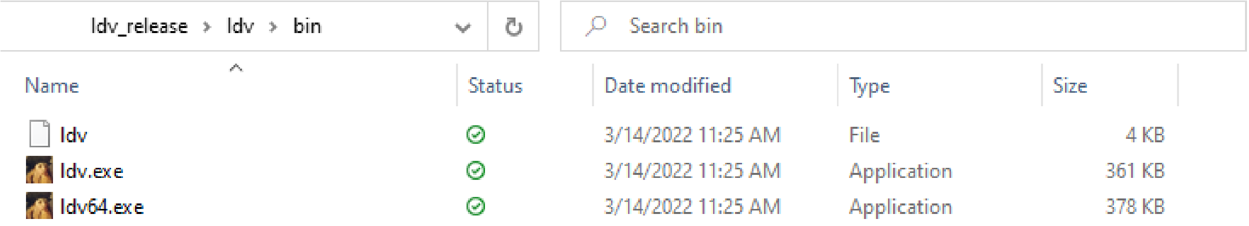


› ldv_release

Name	Status	Date modified	Type	Size
bin	✓	11/21/2020 11:55 AM	File folder	
config	✓	11/21/2020 11:32 AM	File folder	
doc	⟳	11/21/2020 11:32 AM	File folder	
etc	✓	11/21/2020 11:31 AM	File folder	
ldv	✓	11/21/2020 11:31 AM	File folder	
platform	✓	11/21/2020 11:31 AM	File folder	
res	✓	11/21/2020 11:32 AM	File folder	
script	✓	11/21/2020 11:32 AM	File folder	

2.3.3 ldv\bin

This folder contains the application binary executable. Section 3 describes how to launch the software.



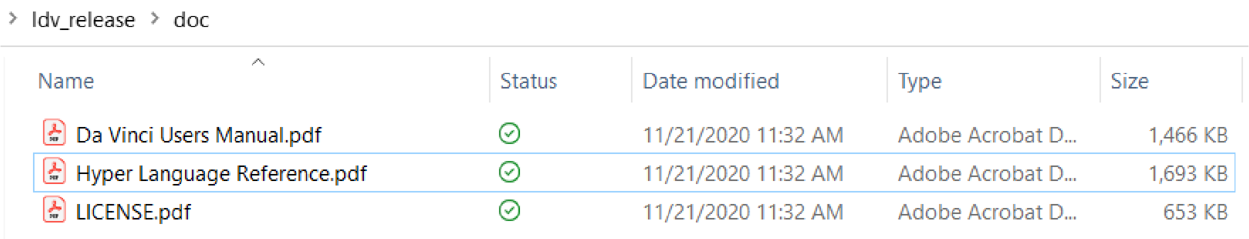
ldv_release › ldv › bin

Search bin

Name	Status	Date modified	Type	Size
ldv	✓	3/14/2022 11:25 AM	File	4 KB
ldv.exe	✓	3/14/2022 11:25 AM	Application	361 KB
ldv64.exe	✓	3/14/2022 11:25 AM	Application	378 KB

2.3.4 ldv\doc

This folder contains documentation, including a user’s manual and a Hyper language reference. These documents can be opened with any PDF reader.



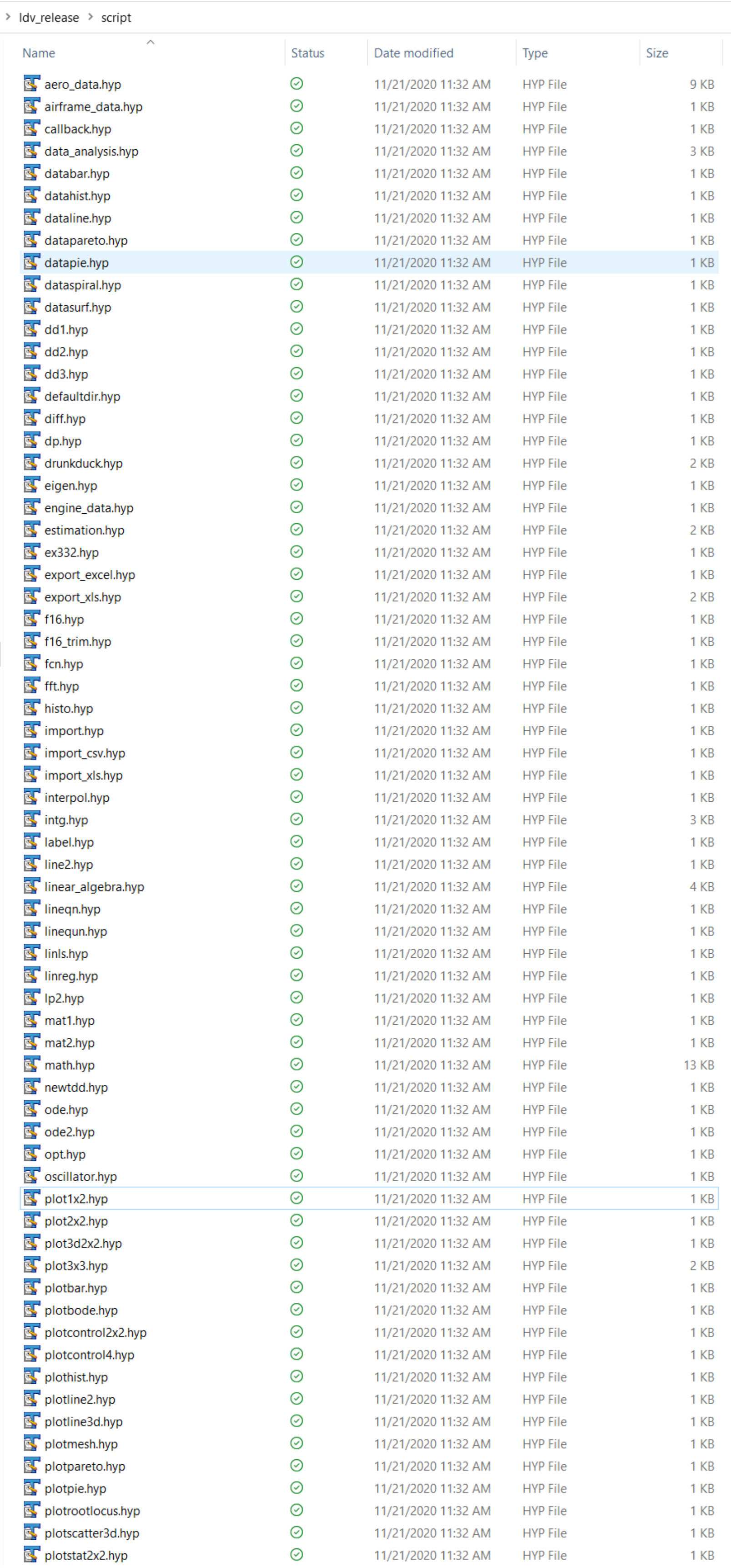
› ldv_release › doc

Name	Status	Date modified	Type	Size
Da Vinci Users Manual.pdf	✓	11/21/2020 11:32 AM	Adobe Acrobat D...	1,466 KB
Hyper Language Reference.pdf	✓	11/21/2020 11:32 AM	Adobe Acrobat D...	1,693 KB
LICENSE.pdf	✓	11/21/2020 11:32 AM	Adobe Acrobat D...	653 KB

2.3.5 ldv\script

This folder contains the scripts that the software can run. This directory can also be accessed through the software itself when browsing for scripts. These scripts are included mostly as examples and to demonstrate various capabilities of the software. More about how to run the scripts and about the scripts is described later in this document.

This is the default folder for running scripts. How to change the default folder is described in the next subsection.

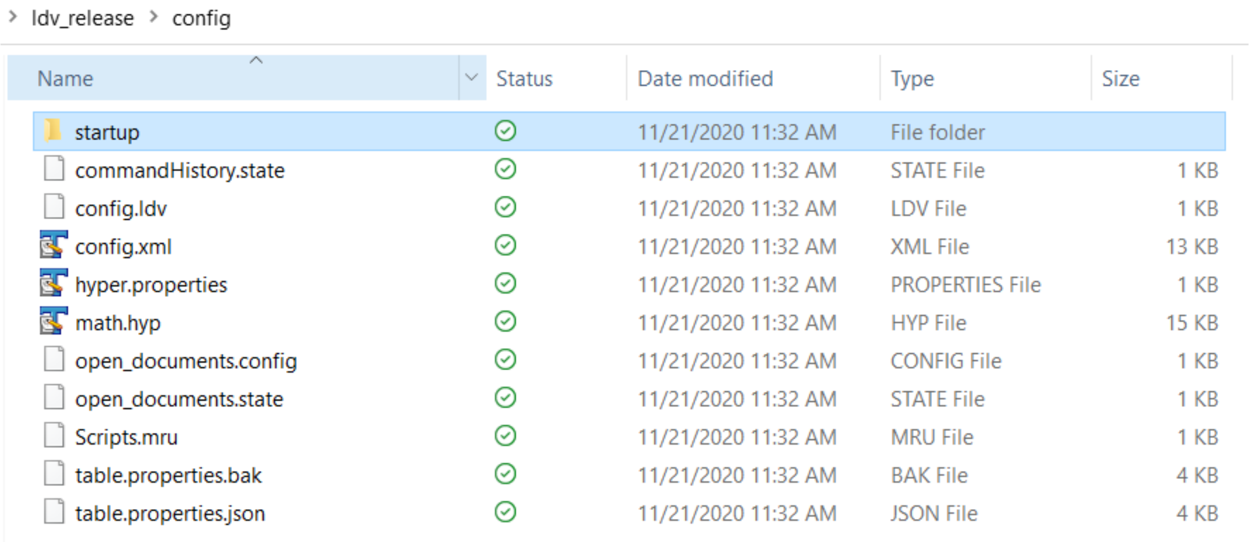


› ldv_release › script

Name	Status	Date modified	Type	Size
aero_data.hyp	✓	11/21/2020 11:32 AM	HYP File	9 KB
airframe_data.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
callback.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
data_analysis.hyp	✓	11/21/2020 11:32 AM	HYP File	3 KB
databar.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
datahist.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
dataline.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
datapareto.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
datapie.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
dataspiral.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
datasurf.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
dd1.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
dd2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
dd3.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
defaultdir.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
diff.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
dp.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
drunkduck.hyp	✓	11/21/2020 11:32 AM	HYP File	2 KB
eigen.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
engine_data.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
estimation.hyp	✓	11/21/2020 11:32 AM	HYP File	2 KB
ex332.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
export_excel.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
export_xls.hyp	✓	11/21/2020 11:32 AM	HYP File	2 KB
f16.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
f16_trim.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
fcn.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
fft.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
histo.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
import.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
import_csv.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
import_xls.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
interpol.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
intg.hyp	✓	11/21/2020 11:32 AM	HYP File	3 KB
label.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
line2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
linear_algebra.hyp	✓	11/21/2020 11:32 AM	HYP File	4 KB
lineqn.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
linequn.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
linls.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
linreg.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
lp2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
mat1.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
mat2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
math.hyp	✓	11/21/2020 11:32 AM	HYP File	13 KB
newtdd.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
ode.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
ode2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
opt.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
oscillator.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plot1x2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plot2x2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plot3d2x2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plot3x3.hyp	✓	11/21/2020 11:32 AM	HYP File	2 KB
plotbar.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotbode.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotcontrol2x2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotcontrol4.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plothist.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotline2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotline3d.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotmesh.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotpareto.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotpie.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotrootlocus.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotscatter3d.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB
plotstat2x2.hyp	✓	11/21/2020 11:32 AM	HYP File	1 KB

2.3.6 ldv\config

This folder contains user preferences and other configuration files.



› ldv_release › config

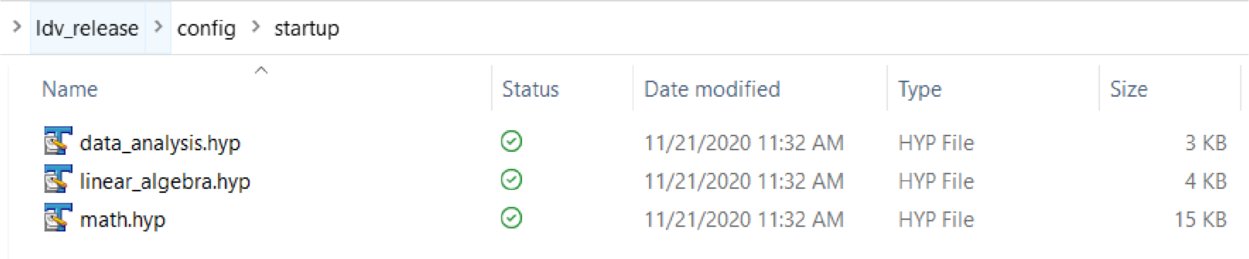
Name	Status	Date modified	Type	Size
startup	✓	11/21/2020 11:32 AM	File folder	
commandHistory.state	✓	11/21/2020 11:32 AM	STATE File	1 KB
config.ldv	✓	11/21/2020 11:32 AM	LDV File	1 KB
config.xml	✓	11/21/2020 11:32 AM	XML File	13 KB
hyper.properties	✓	11/21/2020 11:32 AM	PROPERTIES File	1 KB
math.hyp	✓	11/21/2020 11:32 AM	HYP File	15 KB
open_documents.config	✓	11/21/2020 11:32 AM	CONFIG File	1 KB
open_documents.state	✓	11/21/2020 11:32 AM	STATE File	1 KB
Scripts.mru	✓	11/21/2020 11:32 AM	MRU File	1 KB
table.properties.bak	✓	11/21/2020 11:32 AM	BAK File	4 KB
table.properties.json	✓	11/21/2020 11:32 AM	JSON File	4 KB

To change the default folder, and perform the following steps:

1. open the file config.ldv
2. Uncomment the line that starts with #working_dir by deleting #
3. Edit the path to the desired path.

2.3.7 ldv\config\startup

This folder contains scripts that are run automatically at the startup. Users can add scripts to this folder if they want these scripts to run automatically at the startup.



› ldv_release › config › startup

Name	Status	Date modified	Type	Size
data_analysis.hyp	✓	11/21/2020 11:32 AM	HYP File	3 KB
linear_algebra.hyp	✓	11/21/2020 11:32 AM	HYP File	4 KB
math.hyp	✓	11/21/2020 11:32 AM	HYP File	15 KB

The default directory can also be changed by adding a script in this folder and adding the following command in the script:

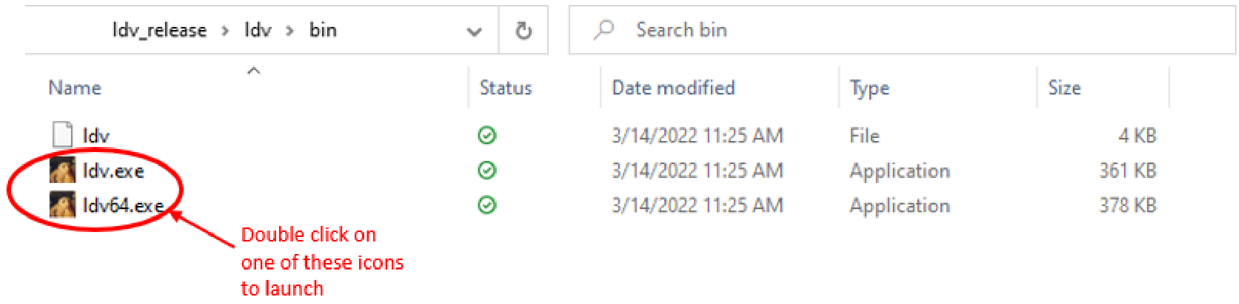
```
changedir(“<desired path>”)
```

2.3.8 Application for Mac

3 Starting

3.1 For PC

1. Navigate to the ldv_release\bin directory.
2. Double click the ldv.exe file to start the application.



Name	Status	Date modified	Type	Size
ldv	✓	3/14/2022 11:25 AM	File	4 KB
ldv.exe	✓	3/14/2022 11:25 AM	Application	361 KB
ldv64.exe	✓	3/14/2022 11:25 AM	Application	378 KB

Double click on one of these icons to launch

3.2 For Mac

Before you can run LDV for Mac, you need to override your security settings and allow the app to install and open by the following the steps below:

Open Finder.

Locate the app you’re trying to open.

Control+Click the app.

Select Open.

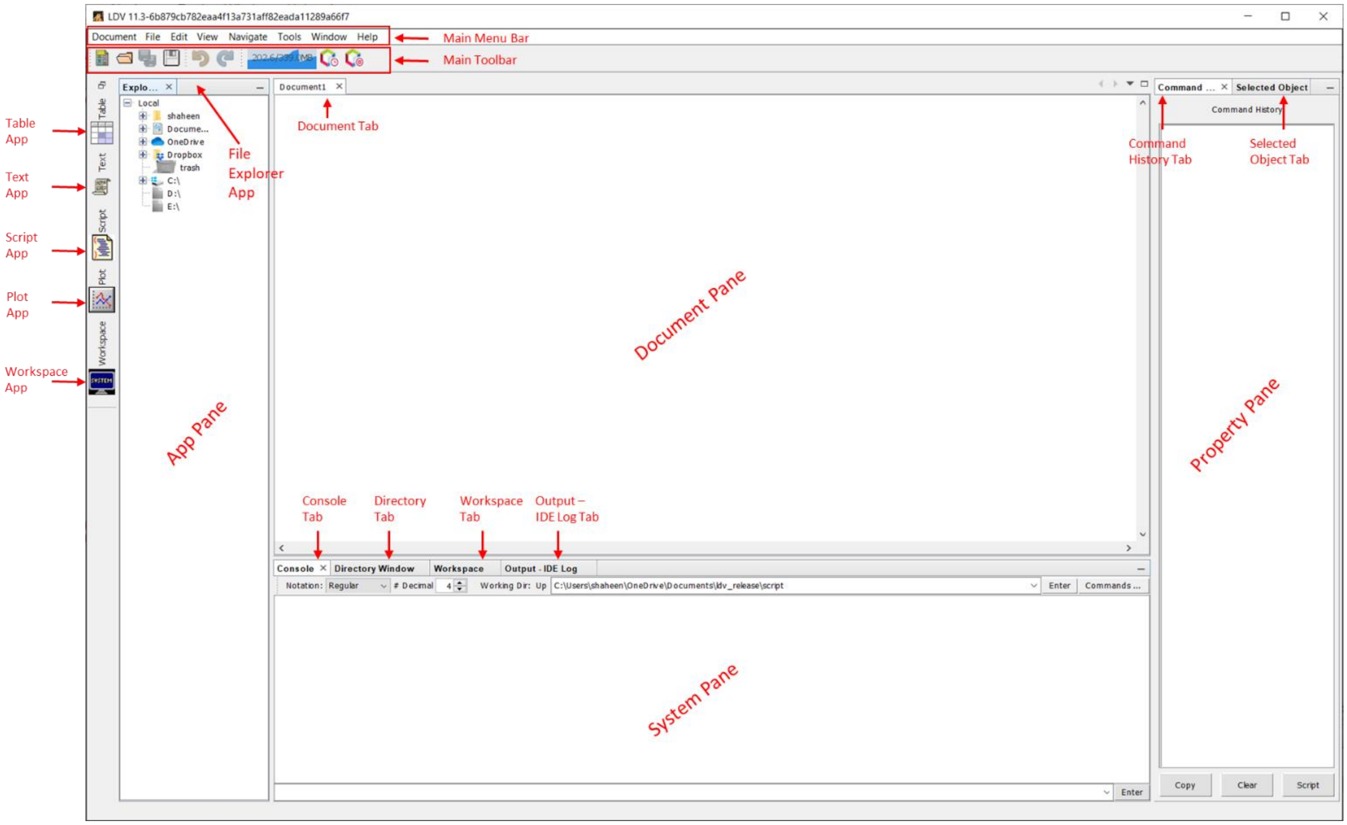
Click Open.

The app should be saved as an exception in your security settings, allowing you to open it in the future.

Ref:

https://www.lifewire.com/fix-developer-cannot-be-verified-error-5183898

4 Graphical User Interface (GUI)

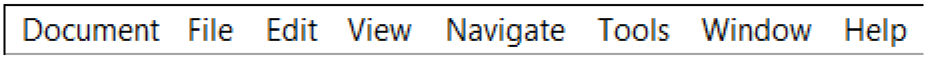


Notation: Regular # Decimal 4 Working Dir: Up C:\Users\shaheen\OneDrive\Documents\ldv_release\script

This is the screen that users should see after launching the software. Like any typical software, LDV has a main menu bar at the top, and a main tool bar right below the menu bar. The GUI is organized with four panes: On the left, there is an App Pane; at the bottom, there is a System Pane; on the right, there is a Property Pane; and at the center top the large pane is the Document Pane. Descriptions of each of the panes is given later in this section.

4.1 Main Menu Bar

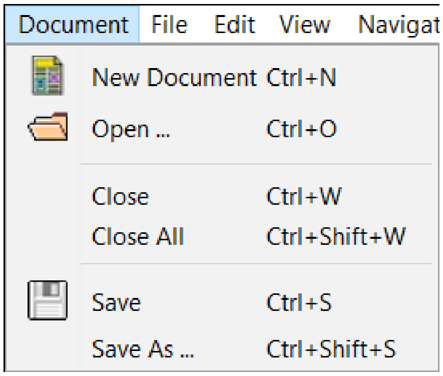
The main menu bar is depicted below:



The main menu bar is located at the top of the GUI. Various functionalities of the software are organized into different menus. Each menu item triggers a utility or a function. Descriptions of the various menus are given below:

Document Menu

The document menu is depicted below:



This menu is used to control documents. It contains options to create, open, close and save documents.

File Menu

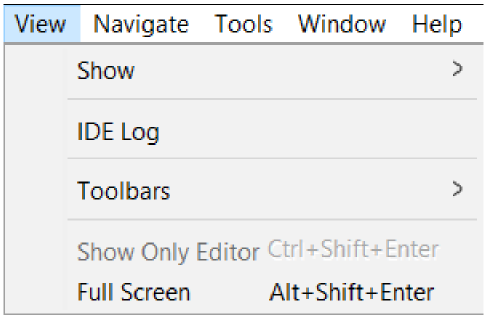
This menu is currently not operational.

Edit Menu

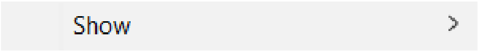
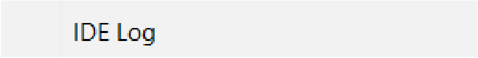
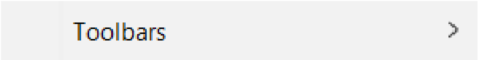
This menu is currently not operational.

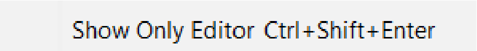
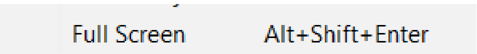
View Menu

The view menu is depicted below:


View
Navigate
Tools
Window
Help
Show >
IDE Log
Toolbars >
Show Only Editor Ctrl+Shift+Enter
Full Screen Alt+Shift+Enter

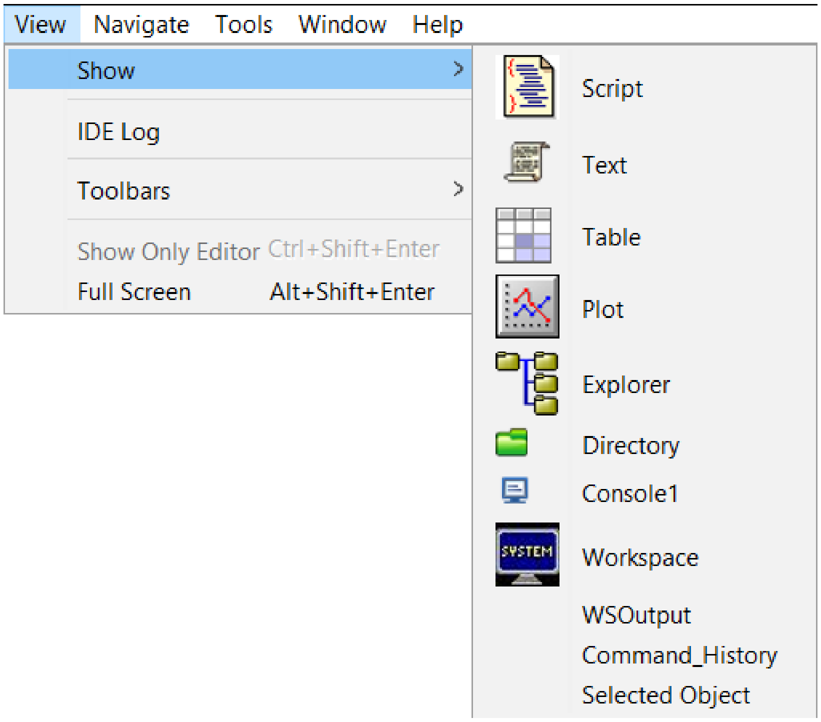
The View menu items are described in the table below:

Menu Item	Description
	Shows or hides various elements, such as different panes, of the software. This submenu is described in more detail in the next subsection.
	Activates the IDE Log tab in the System Pane. IDE Log window displays messages from the Java Runtime Environment (JRE). These messages can be used to diagnose errors in the events of the software malfunction.
	Shows or hides various elements of toolbars such as tools related to files, undo/redo, performance etc.

	This menu item is currently disabled.
	Activates and deactivates the Full Screen Mode for the software. Once in the Full Screen Mode, the menu bar can be made visible by moving the cursor to the top of the screen.

Show Submenu

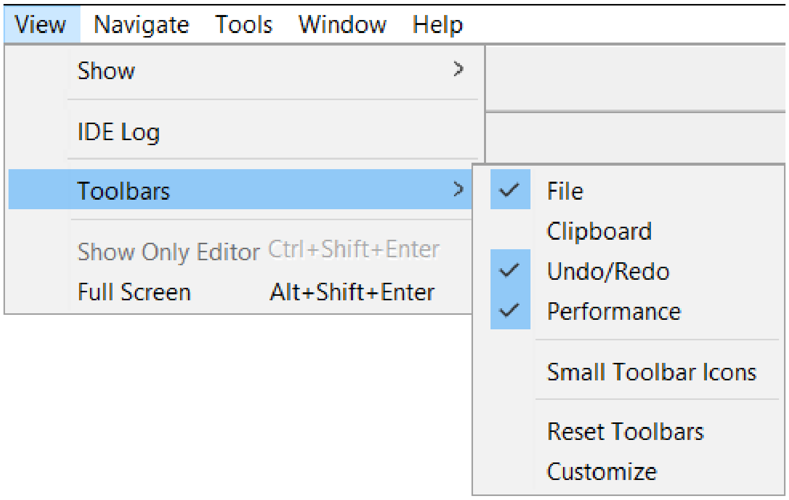
The Show submenu is depicted below:



Selecting each item in the Show submenu will activate the corresponding window.

Toolbars Submenu

The Toolbars submenu is depicted below:



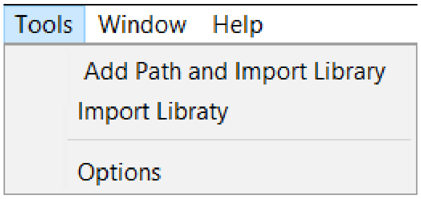
Items in the Toolbars submenu are selected to show or hide different groups of icons in the Main Toolbar.

Navigate Menu

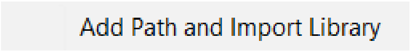
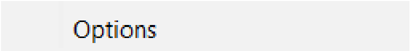
This menu is currently not operational.

Tools Menu

The Tools menu is depicted below:

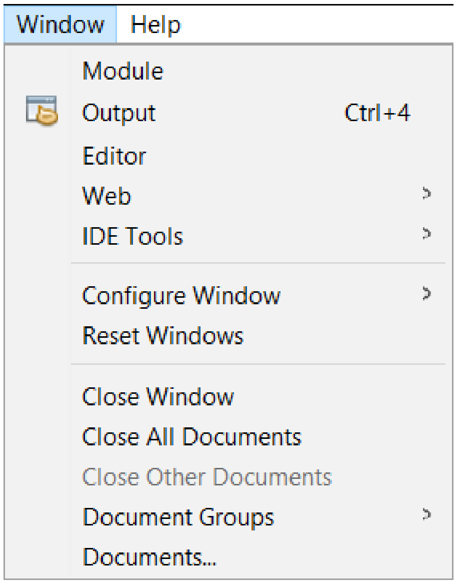


The Tools menu items are described in the table below:

Menu Item	Description
 Add Path and Import Library	opens a dialog box for adding path for a JAR file and importing library. How to import library is described later in this document.
Import Libraty	opens a dialog box for importing library from JAR files that already been loaded. How to import library is described later in this document.
 Options	opens a dialog box for setting options for the application. How to set options is described later in this document.

Window Menu

The Windows menu is depicted below:

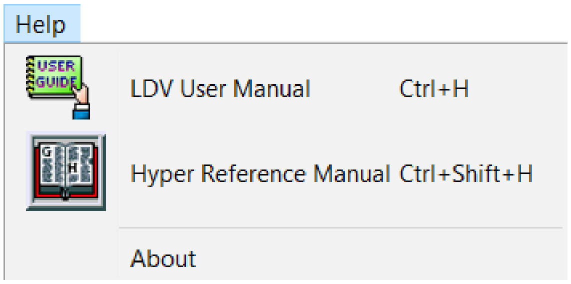


Menu items from this menu are used to manipulate various window. Currently, this menu is not fully functional. This menu will change in the future.

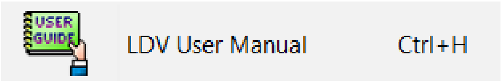
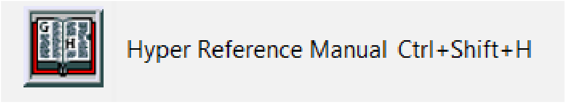
Output Item: This menu item activates the IDE Log tab in the System Pane. IDE Log window displays messages from the Java Runtime Environment (JRE). These messages can be used to diagnose errors in the events of the software malfunction.

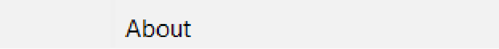
Help Menu

The Help menu is depicted below:



The Help menu items are described in the table below:

Menu Item	Description
 LDV User Manual Ctrl+H	Opens the LDV User's Manual using a default PDF reader. The keyboard short cut is **CRTL+H**.
 Hyper Reference Manual Ctrl+Shift+H	Opens the Hyper Reference Manual using a default PDF reader. The keyboard short cut is **CRTL+Shift+H**.

 About	Opens the splash window for this application, which provides information about the software.

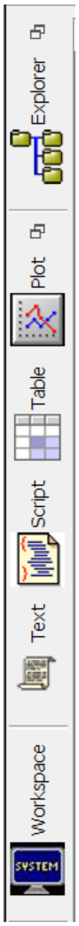
4.2 Main Toolbar

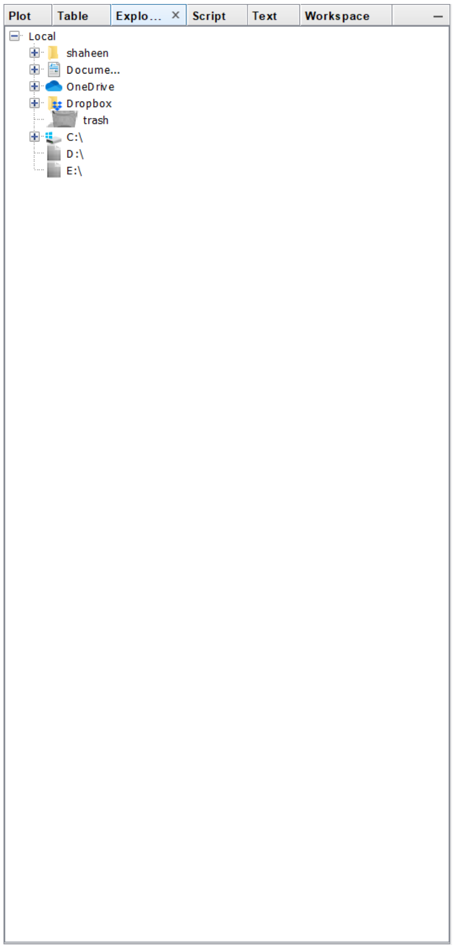
Several of the icons in the Main Toolbar is currently disabled. The Main Toolbar will be updated later.

Icon	Name	Description
	New Document	Open new document for new project
	Open	Opens a document that was previously saved.
	Undo	Currently, disabled.
	Redo	Currently, disabled.
283.9/339.0MB	Memory Usage	Shows the memory uses. Can be used to force the application do garbage collection.
	Profile Application	Profiles the application. It is used to access the performance of the software.
	Pause IO Check	Pauses and resumes the application.

4.3 App Pane

App Pane is used to select tools. A collapse view and an expanded view of the App Pane are depicted below:





Clicking or hovering over each tab exposes the tool pane that contains the tool icons for that app. The tool pane can be docked, undocked, or floated. A floated tool pane can be moved to a different location. This feature is especially useful when using multiple monitors. A tool pane can also be docked at different panes of the application, such as the System Pane, Property Pane, or the Document Pane.

Descriptions of the apps are presented in the table below:

Icon	App Name	Description
	Explorer App	This app is used to navigate the file system of the computer. It is similar to the File Explorer in Window ® and the Finder in Mac OS ®. How to use Explorer App is described later in this document.
	Text App	This app is used to insert texts in a document. It has the similar capabilities of many word processors, such as M.S. Word ®. How to use Text App is described later in this document.
	Table App	This app is used to insert tables in a document. It has similar capabilities of spread sheets, such as M. S. Excel ®. How to use Table App is described later in this document.

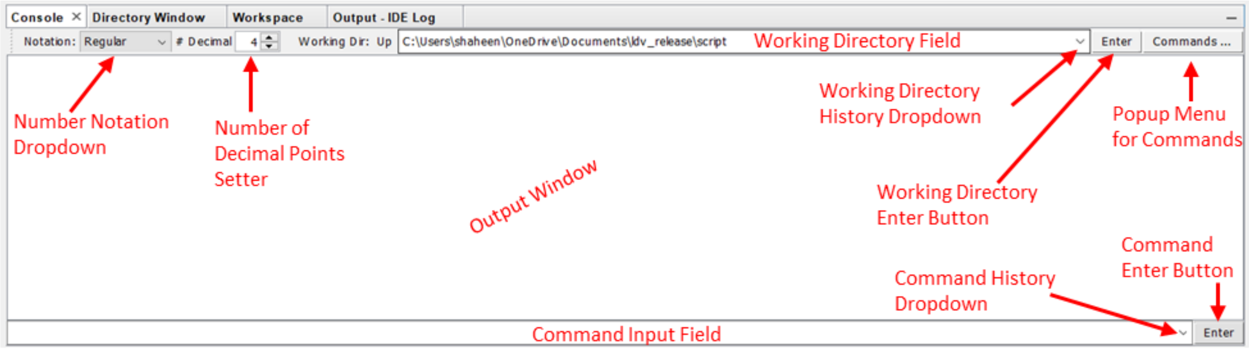
	Script App	This app is used to inset scripts in a document. This app is a script editor, and it has similar capabilities as the editors in many Integrated Development Environments (IDEs), such as Eclipse ®, and Netbeans ®. How to use Script App is described later in this document.
	Plot App	This app is used to insert different types plots in a document. How to use Plot App is described later in this document.
SYSTEM	Workspace App	This is a utility app to explore the workspace. It can be used to check what variables are in the workspace and interrogate the values of the variables. How to use Workspace App is described later in this document.

4.4 System Pane

The System Pane hosts four tabs: Console, Directory, Workspace, and IDE Log. Each tab activates a window.

4.4.1 Console Window

The Console Window is depicted below:



Notation: Regular # Decimal 4 Working Dir: Up C:\Users\shaheen\OneDrive\Documents\ldv_release\script

The top part of the console window features a toolbar. The toolbar contains a number notations drop down menu, a number of decimal point selector, a working directory field, a working directory history dropdown menu, an Enter button, and a button for a popup menu whose items are commonly used commands.

The center part of the Console Window is where the output messages are displayed.

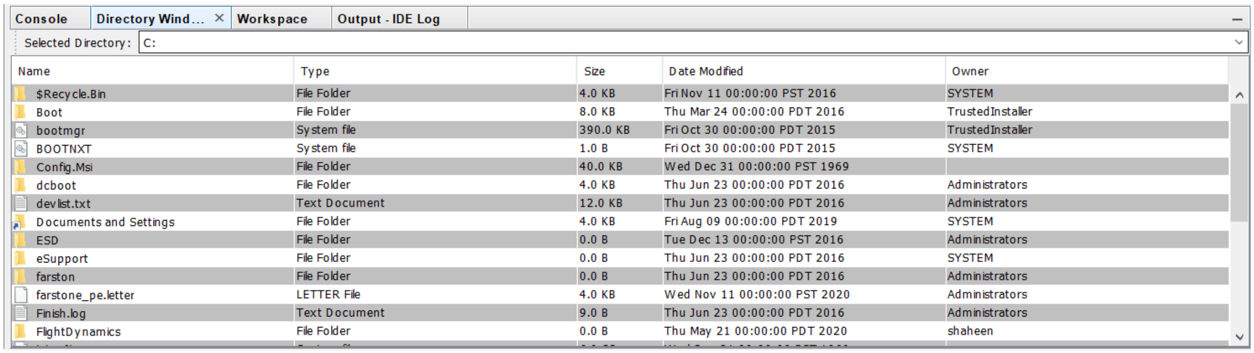
The bottom part of the Console Window consists of a command input field, also known as “command line”, with a command history dropdown, and an Enter button.

There is a Directory tab in the Console Window that shows the directory that the current project is save in.

How to use the Console Window is described later in this document.

4.4.2 Directory Window

The Directory Window is depicted below:



Selected Directory: C:

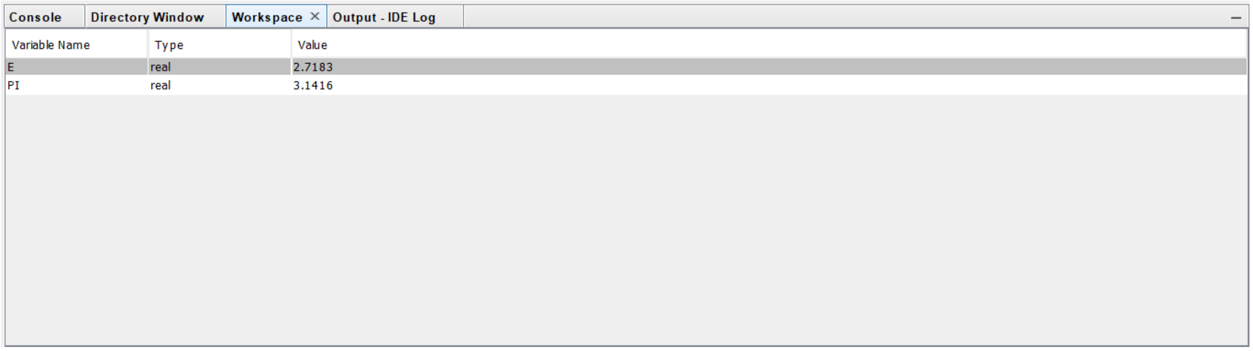
Name	Type	Size	Date Modified	Owner
$Recycle.Bin	File Folder	4.0 KB	Fri Nov 11 00:00:00 PST 2016	SYSTEM
Boot	File Folder	8.0 KB	Thu Mar 24 00:00:00 PDT 2016	TrustedInstaller
bootmgr	System file	390.0 KB	Fri Oct 30 00:00:00 PDT 2015	TrustedInstaller
BOOTNXT	System file	1.0 B	Fri Oct 30 00:00:00 PDT 2015	SYSTEM
Config.Msi	File Folder	40.0 KB	Wed Dec 31 00:00:00 PST 1969	
dcboot	File Folder	4.0 KB	Thu Jun 23 00:00:00 PDT 2016	Administrators
devlist.txt	Text Document	12.0 KB	Thu Jun 23 00:00:00 PDT 2016	Administrators
Documents and Settings	File Folder	4.0 KB	Fri Aug 09 00:00:00 PDT 2019	SYSTEM
ESD	File Folder	0.0 B	Tue Dec 13 00:00:00 PST 2016	Administrators
eSupport	File Folder	0.0 B	Thu Jun 23 00:00:00 PDT 2016	SYSTEM
farston	File Folder	0.0 B	Thu Jun 23 00:00:00 PDT 2016	Administrators
farstone_pe.letter	LETTER File	4.0 KB	Wed Nov 11 00:00:00 PST 2020	Administrators
Finish.log	Text Document	9.0 B	Thu Jun 23 00:00:00 PDT 2016	Administrators
FlightDynamics	File Folder	0.0 B	Thu May 21 00:00:00 PDT 2020	shaheen

The Directory Window works with the Explorer App. The top part of the Directory Window contains a field that displays the selected directory in the Explorer App. The bottom part of the Directory Window displays the contents of the selected directory.

How to use the Directory Window is described later in this document.

4.4.3 Workspace Window

The Workspace window is depicted below:



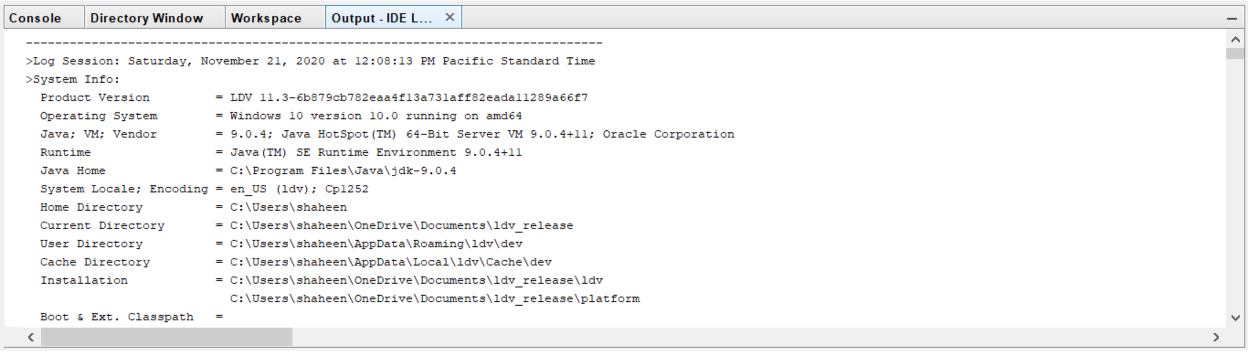
Variable Name	Type	Value
E	real	2.7183
PI	real	3.1416

This window displays the variables and their types and contents selected in the Workspace app. This window contains a table with three columns. The column headers are Variable Name, Type, and Value.

How to use the Directory Window is described later in this document.

4.4.4 IDE Log Window

The IDE Log window is depicted below:



```
------------------------------------------------------------------------------
>Log Session: Saturday, November 21, 2020 at 12:08:13 PM Pacific Standard Time
>System Info:
  Product Version         = LDV 11.3-6b879cb782eaa4f13a731aff82eada11289a66f7
  Operating System        = Windows 10 version 10.0 running on amd64
  Java; VM; Vendor        = 9.0.4; Java HotSpot(TM) 64-Bit Server VM 9.0.4+11; Oracle Corporation
  Runtime                 = Java(TM) SE Runtime Environment 9.0.4+11
  Java Home               = C:\Program Files\Java\jdk-9.0.4
  System Locale; Encoding = en_US (ldv); Cp1252
  Home Directory          = C:\Users\shaheen
  Current Directory       = C:\Users\shaheen\OneDrive\Documents\ldv_release
  User Directory          = C:\Users\shaheen\AppData\Roaming\ldv\dev
  Cache Directory         = C:\Users\shaheen\AppData\Local\ldv\Cache\dev
  Installation            = C:\Users\shaheen\OneDrive\Documents\ldv_release\ldv
                            C:\Users\shaheen\OneDrive\Documents\ldv_release\platform
  Boot & Ext. Classpath   =
```

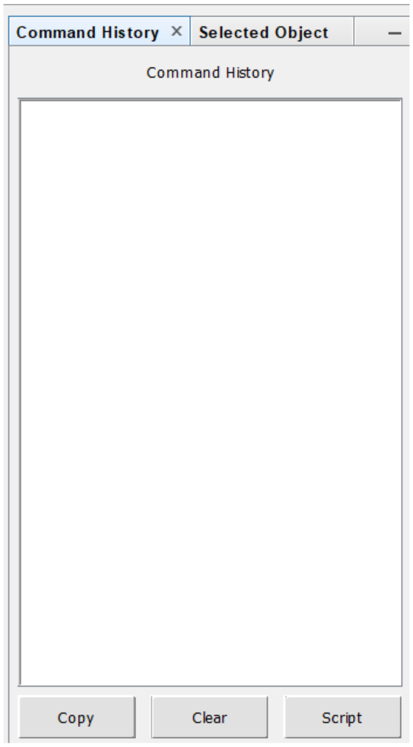
This window displays the Java output of the application.

4.5 Properties Pane

The Properties pane currently hosts two tabs: Command History and Selected Object.

4.5.1 Command History

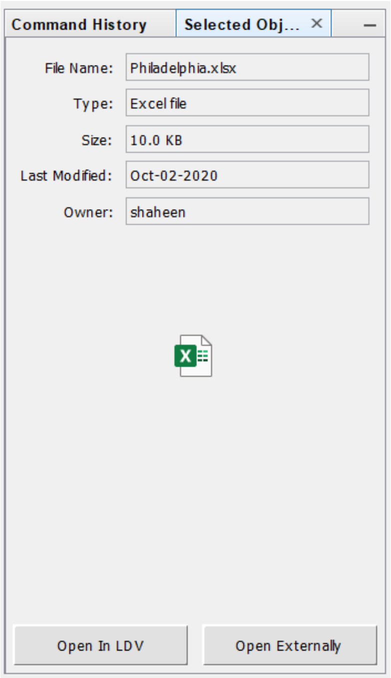
The Command History tab activates a window that displays command history. The Command History tab is depicted below:



4.5.2 Selected Object

The Selected Object tab activates a pane that contains the properties of an object that is selected either in the System pane or in the Document pane. The view for properties of each object is customized according to the object selected. In general, these panes display properties of the selected object and allows to perform some action on the selected object. More about their functionalities are described in the How to Use section of this document. Several examples are given below:

Selected file:



File Name: Philadelphia.xlsx

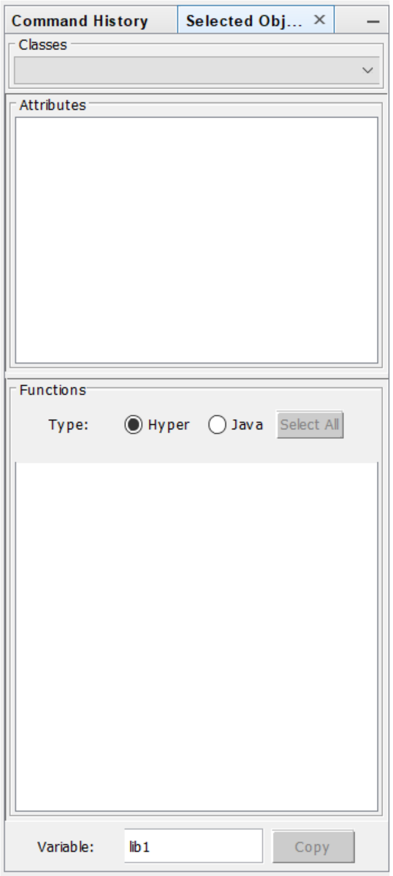
Type: Excel file

Size: 10.0 KB

Last Modified: Oct-02-2020

Owner: shaheen

The figure above shows a properties window for a selected Excel file. In addition to displaying the properties, there are two buttons which allow the selected file to be opened either externally in the native application of the file or internally inside LDV.

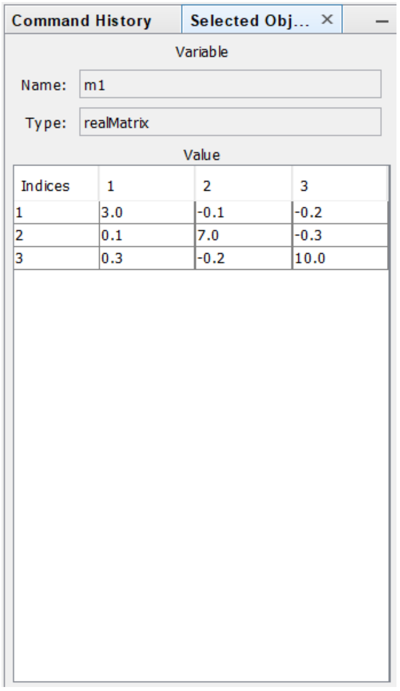


Type: ● Hyper ○ Java

Variable: lib1

The figure above shows the properties pane of a JAR file.

Selected Variable:



Variable

Name: m1

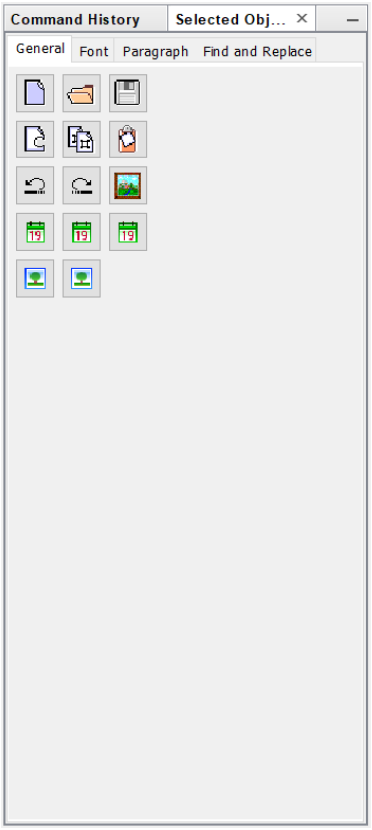
Type: realMatrix

Value

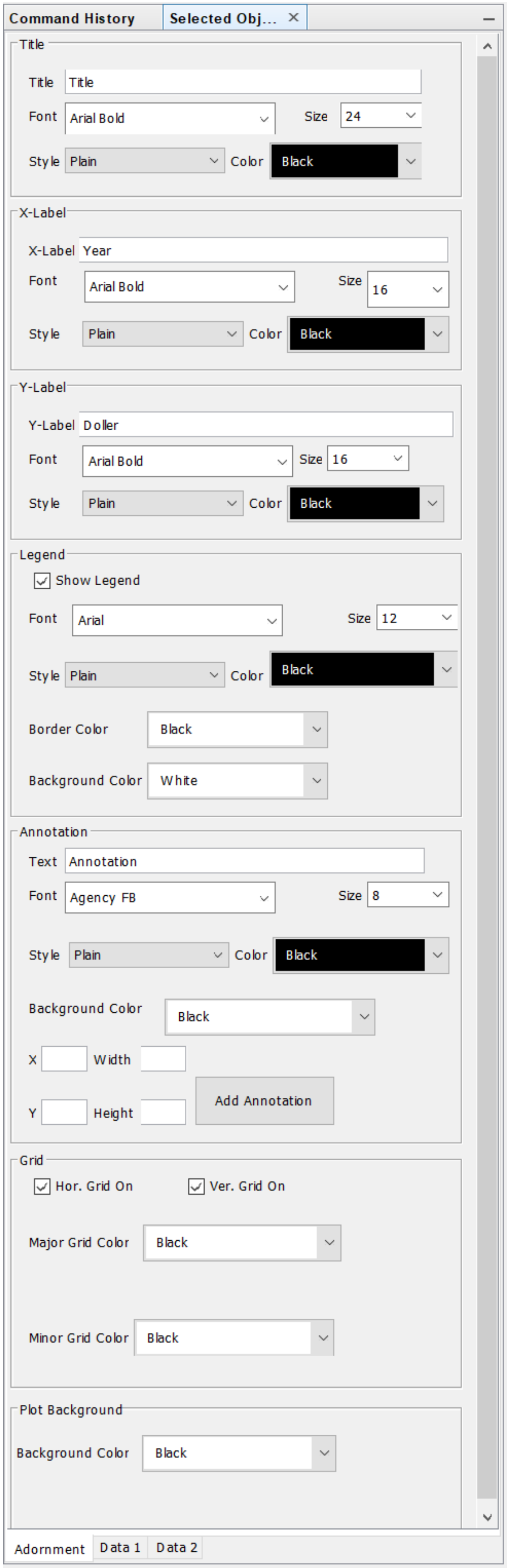
Indices	1	2	3
1	3.0	-0.1	-0.2
2	0.1	7.0	-0.3
3	0.3	-0.2	10.0

The figure above shows a properties pane for a selected variable. In addition to displaying the name and the type of the variable, it also displays the value of the variable.

Selected Text Box:



The figure above shows the properties pane of a text box selected in a document. The properties are organized by tabs. Each tab displays subset of the related properties grouped together. Most of these properties can be manipulated from the properties pane.



Title

Title: Title
Font: Arial Bold
Size: 24
Style: Plain
Color: Black

X-Label

X-Label: Year
Font: Arial Bold
Size: 16
Style: Plain
Color: Black

Y-Label

Y-Label: Doller
Font: Arial Bold
Size: 16
Style: Plain
Color: Black

Legend

☑ Show Legend
Font: Arial
Size: 12
Style: Plain
Color: Black
Border Color: Black
Background Color: White

Annotation

Text: Annotation
Font: Agency FB
Size: 8
Style: Plain
Color: Black
Background Color: Black
X
Width
Y
Height

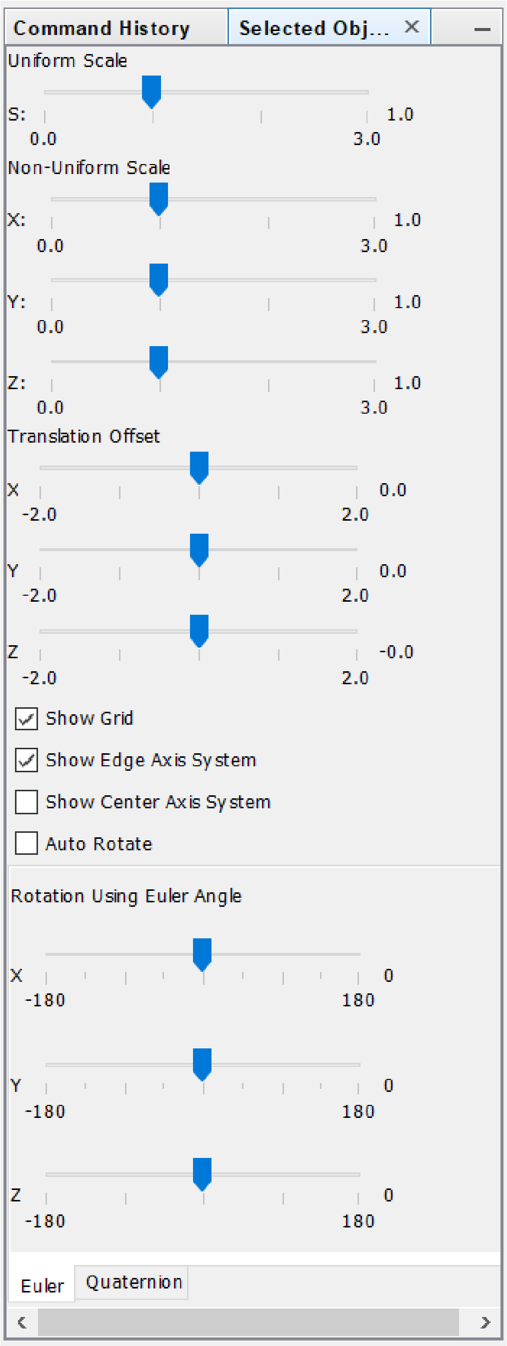
Grid

☑ Hor. Grid On
☑ Ver. Grid On
Major Grid Color: Black
Minor Grid Color: Black

Plot Background

Background Color: Black

The figure above shows a properties pane of a selected 2D plot. The properties are organized by tabs. Each tab displays subset of the related properties grouped together. Most of these properties can be manipulated from the properties pane.



Uniform Scale

S: 0.0 – 3.0 1.0

Non-Uniform Scale

X: 0.0 – 3.0 1.0

Y: 0.0 – 3.0 1.0

Z: 0.0 – 3.0 1.0

Translation Offset

X -2.0 – 2.0 0.0

Y -2.0 – 2.0 0.0

Z -2.0 – 2.0 -0.0

☑ Show Grid

☑ Show Edge Axis System

☐ Show Center Axis System

☐ Auto Rotate

Rotation Using Euler Angle

X -180 – 180 0

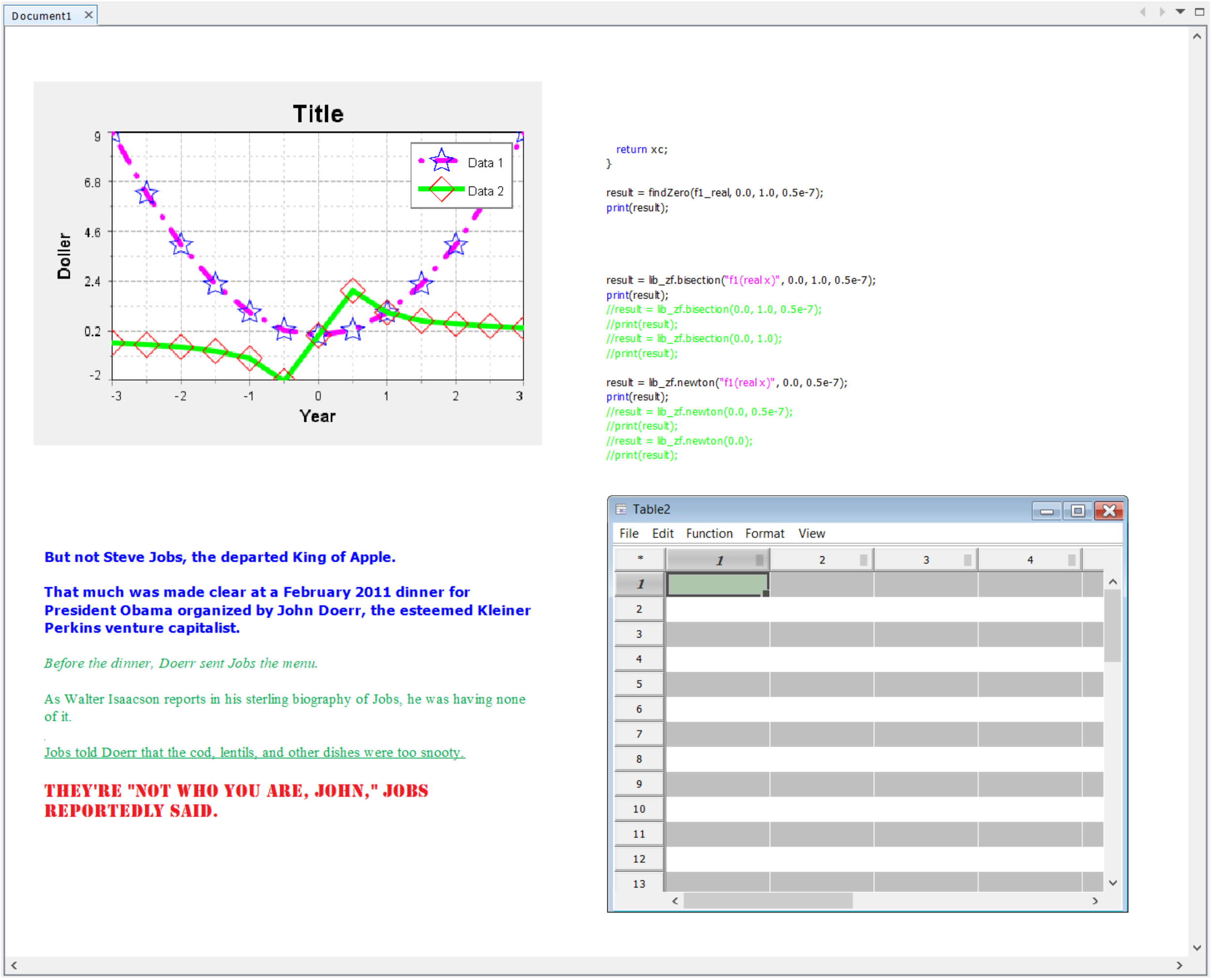
Y -180 – 180 0

Z -180 – 180 0

The figure above shows a properties pane of a selected 3D plot. The properties are organized by tabs. Each tab displays subset of the related properties grouped together. Most of these properties can be manipulated from the properties pane.

4.6 Document Plane

The Document pane hosts documents in separate tabs. LDV documents are universal documents, meaning that they can contains different types of data. An example of a typical document is shown below:



```
  return xc;
}

result = findZero(f1_real, 0.0, 1.0, 0.5e-7);
print(result);



result = lib_zf.bisection("f1(real x)", 0.0, 1.0, 0.5e-7);
print(result);
//result = lib_zf.bisection(0.0, 1.0, 0.5e-7);
//print(result);
//result = lib_zf.bisection(0.0, 1.0);
//print(result);

result = lib_zf.newton("f1(real x)", 0.0, 0.5e-7);
print(result);
//result = lib_zf.newton(0.0, 0.5e-7);
//print(result);
//result = lib_zf.newton(0.0);
//print(result);
```

But not Steve Jobs, the departed King of Apple.

That much was made clear at a February 2011 dinner for President Obama organized by John Doerr, the esteemed Kleiner Perkins venture capitalist.

Before the dinner, Doerr sent Jobs the menu.

As Walter Isaacson reports in his sterling biography of Jobs, he was having none of it.

Jobs told Doerr that the cod, lentils, and other dishes were too snooty.

THEY'RE "NOT WHO YOU ARE, JOHN," JOBS REPORTEDLY SAID.

Documents are containers of components provided by various apps. Components produced by different apps are of different data types. The above figure shows a plot, a script, a text box, and a table. Only one of the components can be active at any time. The active component is adorned by a frame, a menu bar, and/or a toolbar. The inactive frames are embedded in the document. In the figure above, an active Table component with frame is shown.

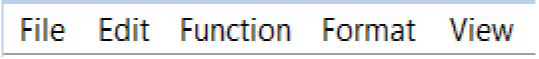
4.6.1 App Frames

Components created by the Apps are contained in the App frames. App frames and data components are embedded in Documents. App contents can be created using either GUI or commands (via command line or script). After selecting the **[Insert]** button of a particular app from its Tool pane, clicking in a document creates a frame with predefined size and clicking plus dragging creates a frame of desired size. Clicking outside the frame embeds the content of the frame into the document. There are elements common to all App Frames, such as menu bars and toolbars. Menu bars also contains menus that are common to all Apps. Functions that are common to all apps, such as open, save, etc., are implemented using common menu items and common tool icons in the toolbar. Specific menu items and tool icons are added to different app menus and toolbars that represents specific functions of those apps.

Each embedded item in a document is called an object. Each object has a handler (pointer or reference). The handlers are used to send commands to the objects from the Command Line or scripts. The name of the frame is the handler for that object. The name is originally assigned at the time of the object creation, and the object can be renamed after their creation.

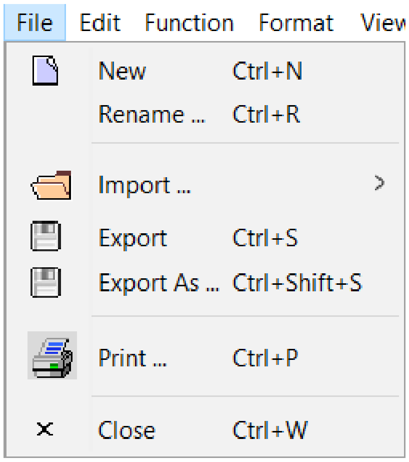
App Menu Bar

An app menu bar is depicted below:



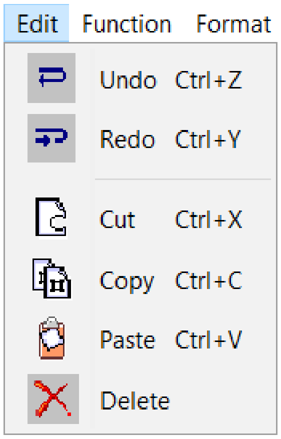
File Menu:

A File menu of an app menu bar is depicted below:



Edit Menu:

An Edit menu of an app menu bar is depicted below:



Functions Menu:

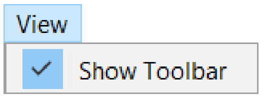
A Function menu contains functions that are specific to an app; therefore, there are no common menu items.

Format Menu:

A Format menu contains functions that are specific to an app; therefore, there are no common menu items.

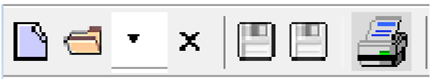
View Menu:

A View menu of an app menu bar is depicted below:



App Toolbar

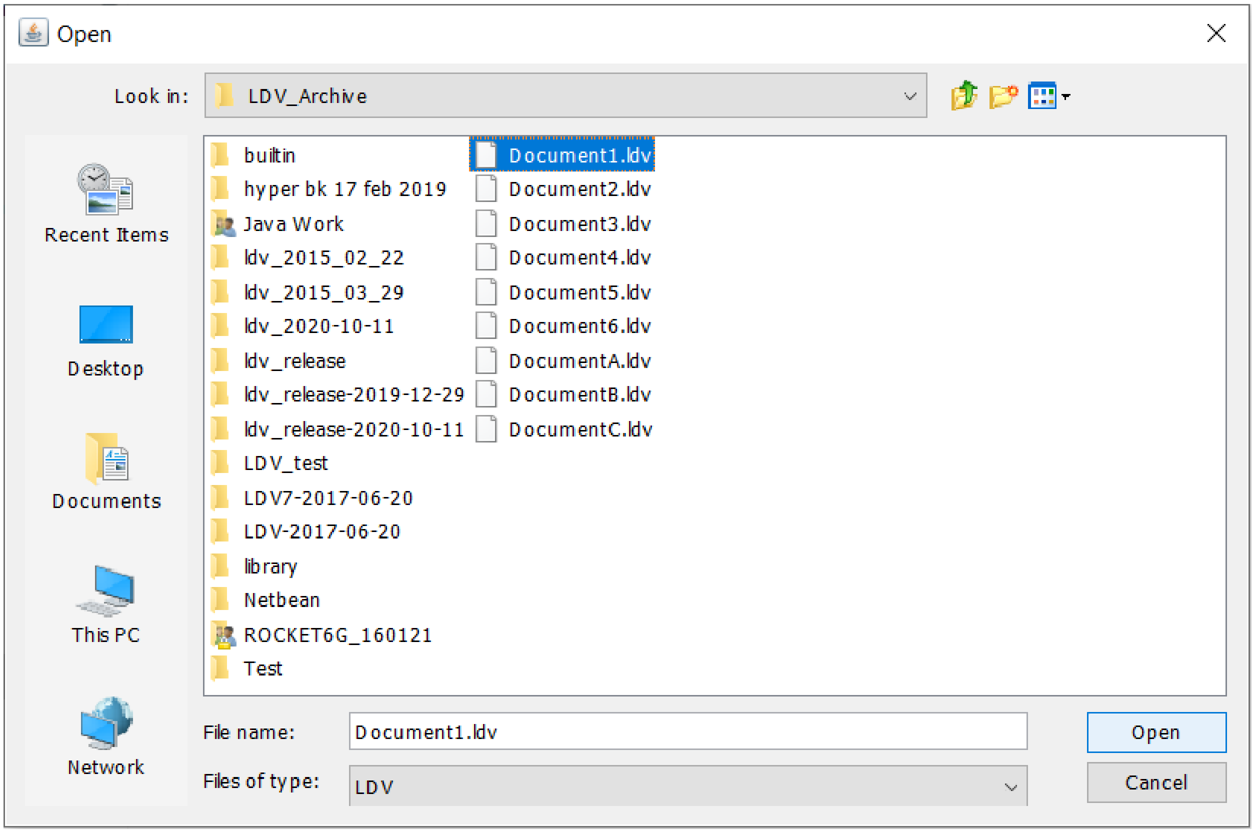
The toolbar on the app frame is invisible by default. It can be made visible from the View menu by checking View > Toolbar. A common app toolbar is depicted below:



4.7 Dialog Boxes

Dialog boxes are opened when certain functions are activated, such as Open (Import), close (Export), Print, etc. Various dialog boxes are shown below:

Open Dialog Box:

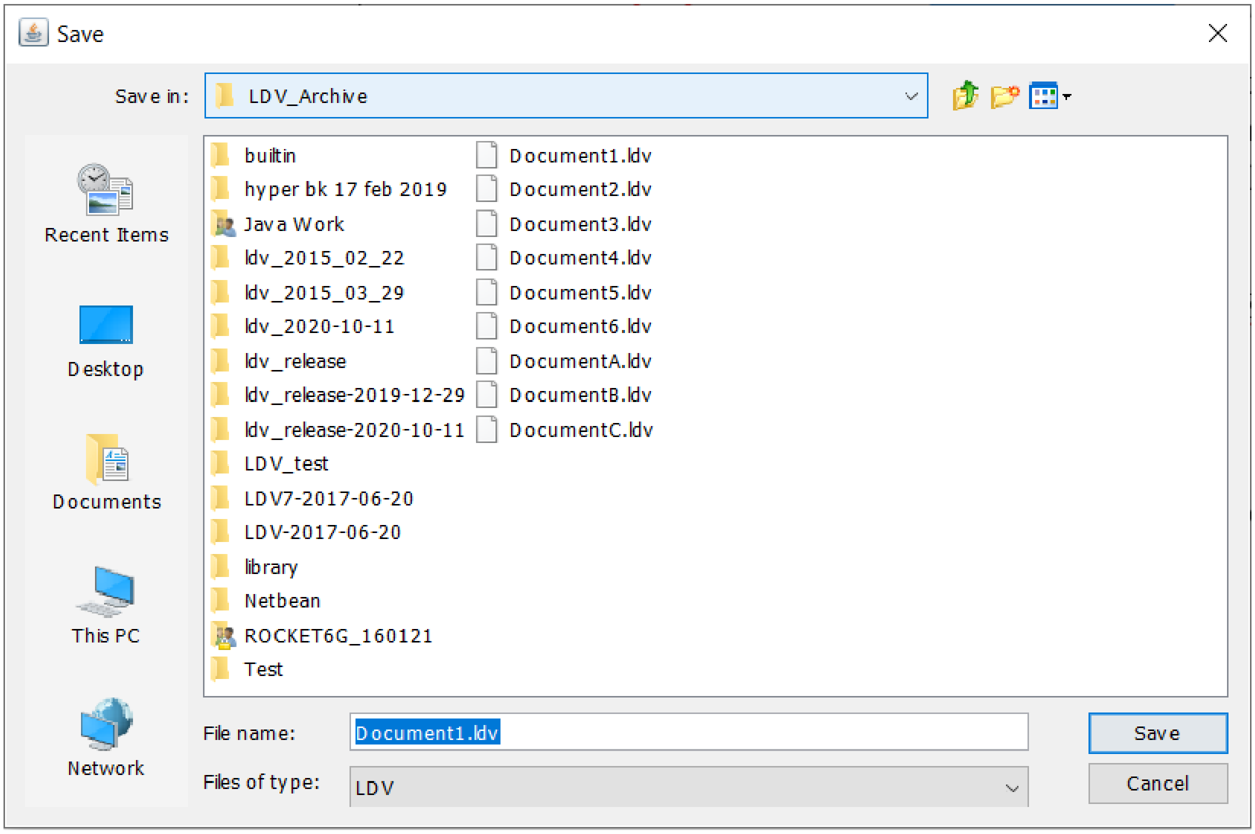


Look in: LDV_Archive

File name: Document1.ldv

Files of type: LDV

Save Dialog Box:

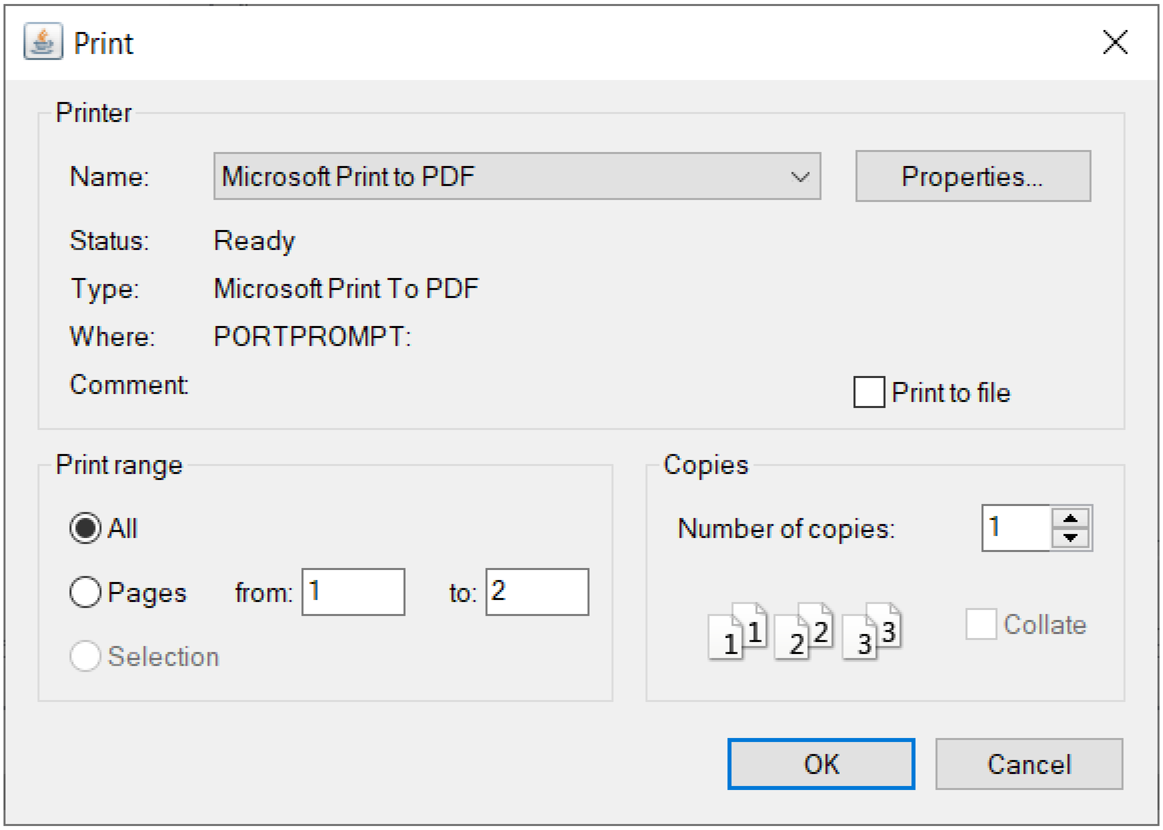


Save in: LDV_Archive

File name: Document1.ldv

Files of type: LDV

Print Dialog Box:



Printer

Name: Microsoft Print to PDF

Status: Ready

Type: Microsoft Print To PDF

Where: PORTPROMPT:

Comment:

□ Print to file

Print range

● All

○ Pages from: 1 to: 2

○ Selection

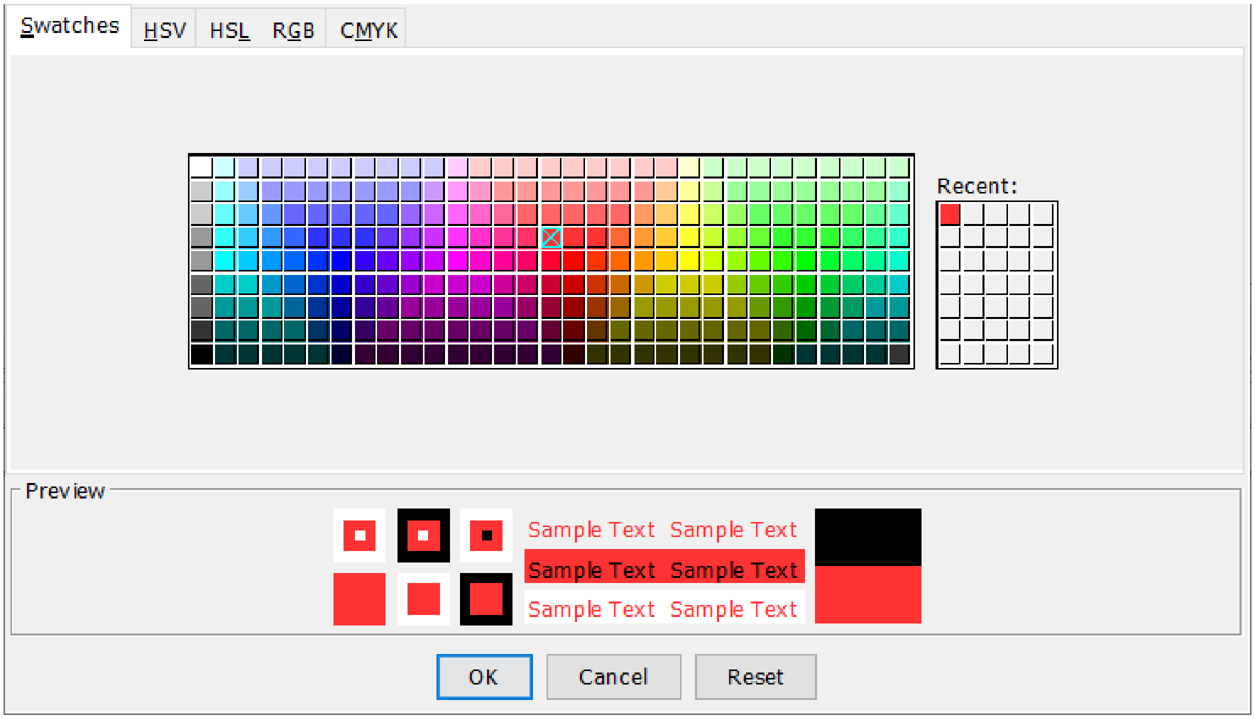
Copies

Number of copies: 1

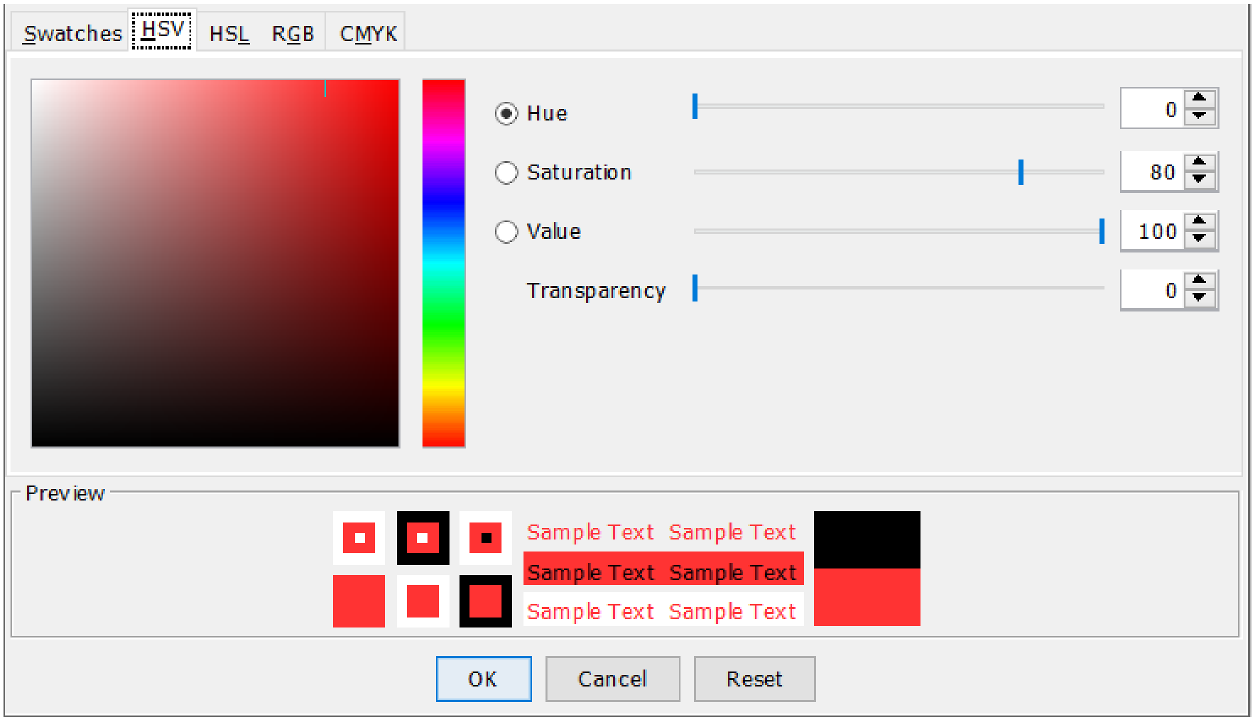
□ Collate

Color Selection Dialog Box:

The color selection dialog box has five tabs for different color selection methods. The first tab, which has the color swatches is shown below:



The second tab, which has the Hue, Saturation, and Value (HSV) color model is shown below:



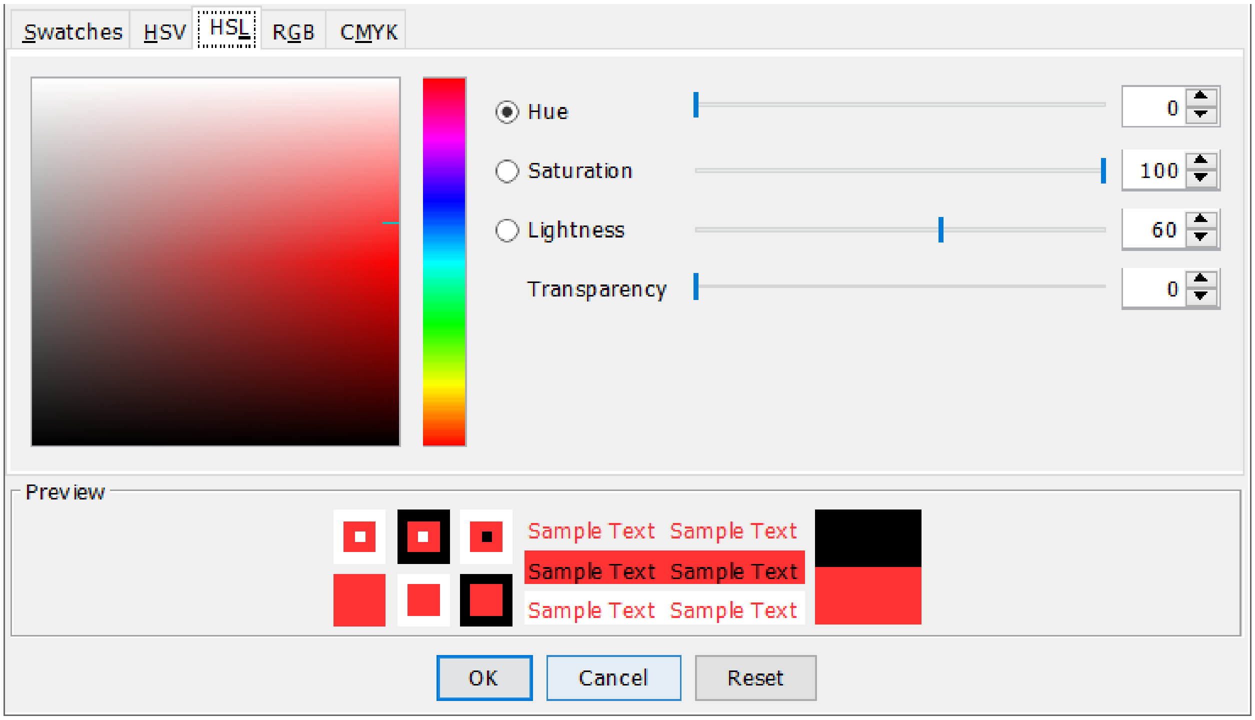
● Hue 0

○ Saturation 80

○ Value 100

Transparency 0

The third tab, which has the Hue Saturation, and Lightness (HSL) color model is shown below:



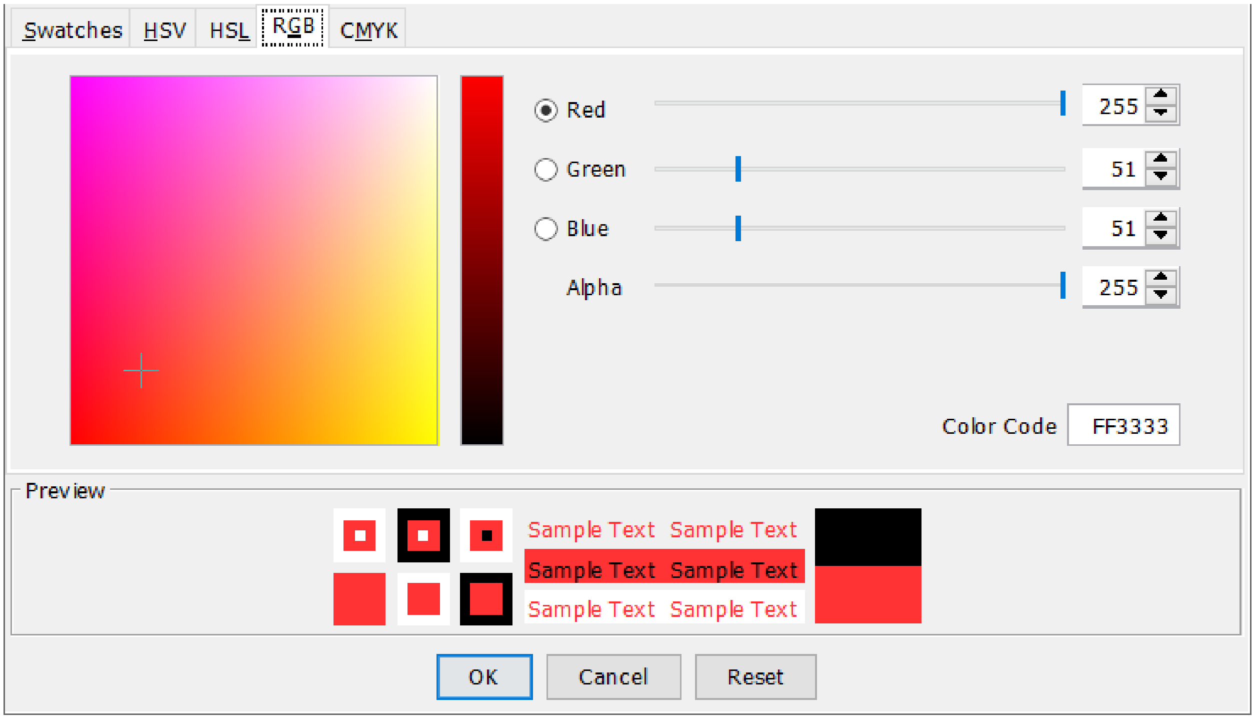
● Hue 0

○ Saturation 100

○ Lightness 60

Transparency 0

The fourth tab, which has the Red, Green, and Blue (RGB) color model is shown below:



● Red 255

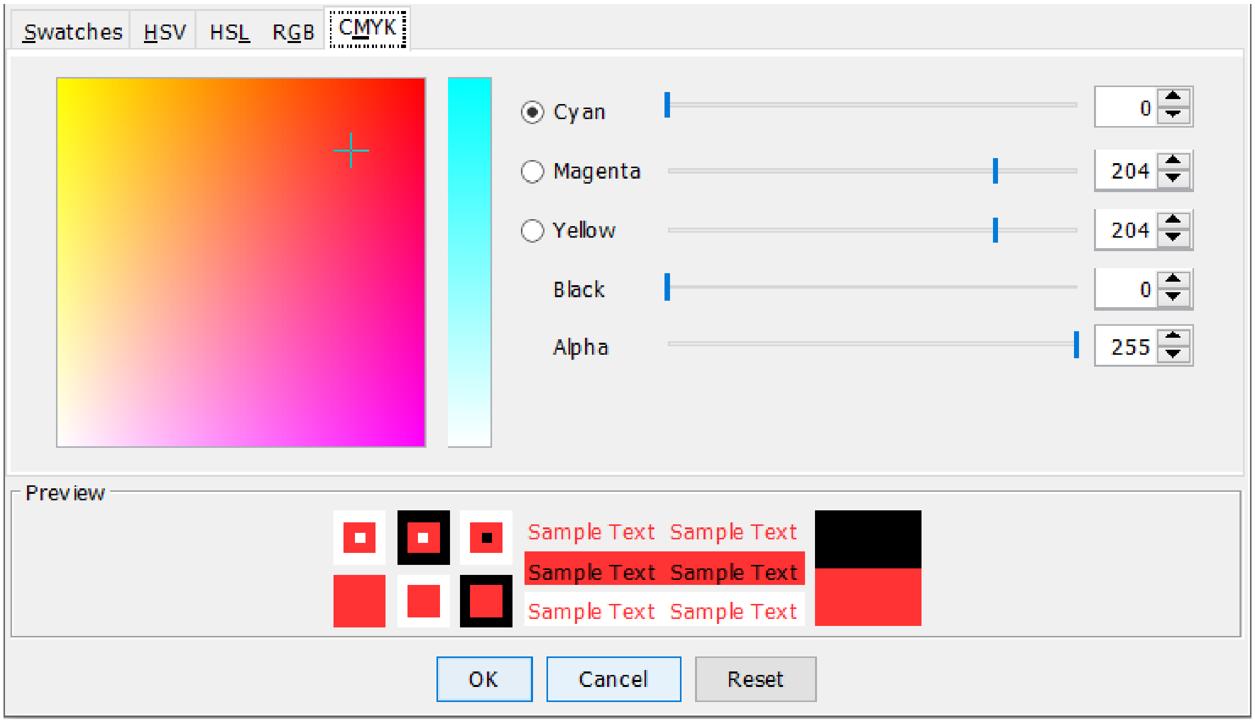
○ Green 51

○ Blue 51

Alpha 255

Color Code FF3333

The fifth tab, which has the Cyan, Yellow, Magenta, and Black (CYMK) color model is shown below:



- ● Cyan 0
- ○ Magenta 204
- ○ Yellow 204
- Black 0
- Alpha 255

After selecting the desired color, clicking the [OK] button applies the selected color to the selected object and dismisses the dialog box. Clicking on the [Cancel] button cancels the operation and dismisses the dialog box. Clicking on the [Reset] button sets all the color values to zero.

5 Text User Interface (TUI)

In addition to having a Graphical User Interface (GUI), LDV also has a Text User Interface (TUI). The TUI is implemented through the Console, the Command Line, and Script. The syntax for the TUI is documented in the Hyper Reference manual.

Some TUI statements are applicable to the overall LDV application, and some TUI statements are specific to an app. The app statements are described in the section of the relevant app. The LDV statements are described in the table below:

Function	Description
`openFile(String path)`	Opens a file specified by the parameter path. Based on the extension of the file, the file can be opened as a document, inside an app frame, or with an external application.
`importFile(String path)`	Imports a file specified by the parameter path inside an app frame.

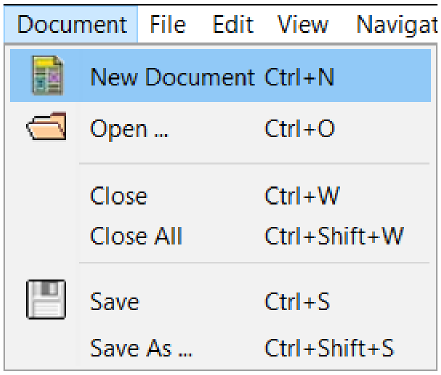
6 Using LDV

6.1 Working with Document

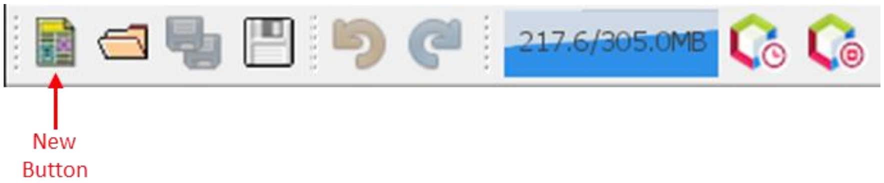
In LDV, components from various apps are composed in documents. LDV documents are containers of components provided by various apps. Different apps produce different data types.

6.1.1 New Document

When LDV is launched, a new document is automatically created. Additional documents can be created by either selecting the menu item **[New Document]** from the **Document** menu in the main menu bar as shown below



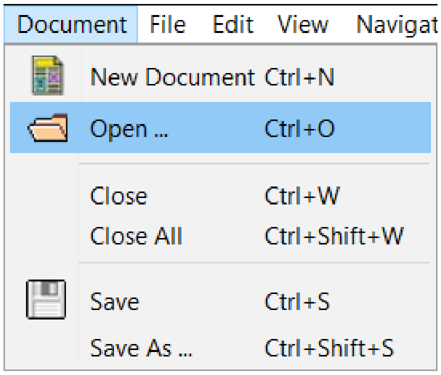
or by clicking the **[New]** button on the main toolbar as shown below.



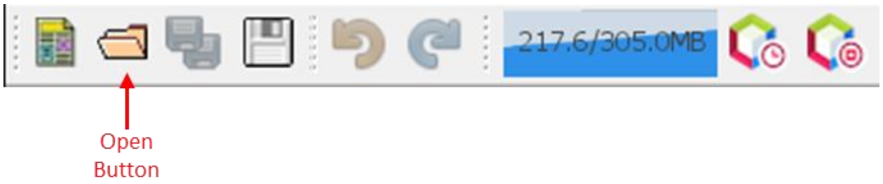
A new document can also be created by using the keyboard shortcut **CTRL+N**.

6.1.2 Opening Document

An existing document can be opened by either selecting the menu item **[Open …]** from the **Document** menu in the main menu bar as shown below



or by clicking the **[Open]** button on the main toolbar as sown below.



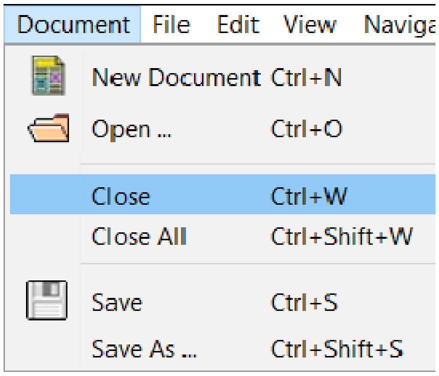
An existing document can also be opened by using the keyboard shortcut **CTRL+O**.

Choosing one of the options above will open a dialog box as shown in Section 4.7.

The dialog box is used to locate the existing document file. LDV documents have extension **`ldv`**. Once located, the file is selected and clicking on the **[Open]** button will open the selected document. Clicking on the **[Cancel]** button cancels opening document.

6.1.3 Closing Document

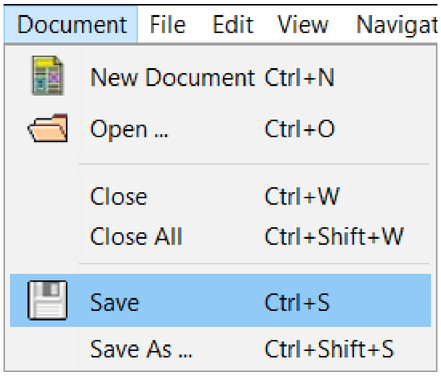
An opened document is closed by making the document tab active and selecting the menu item **[Close]** from the **Document** menu in the main menu bar as shown below.



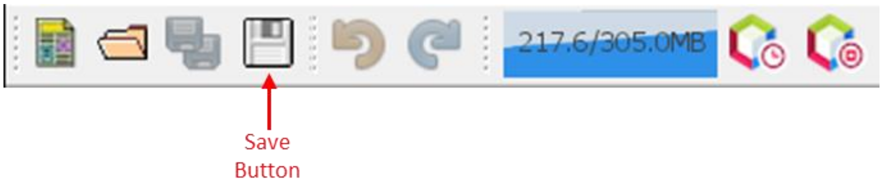
The selected document can also be closed by using the keyboard shortcut **CTRL+W**. Selecting the menu item **[Close All]** or using the keyboard shortcut **CTRL+Shift+W** closes all the open documents.

6.1.4 Saving Document

An opened and modified document is saved by making the document tab active and either selecting the menu item **[Save]** from the **Document** menu in the main menu bar, as shown below,



or by clicking the **[Save]** button on the main toolbar, as sown below.



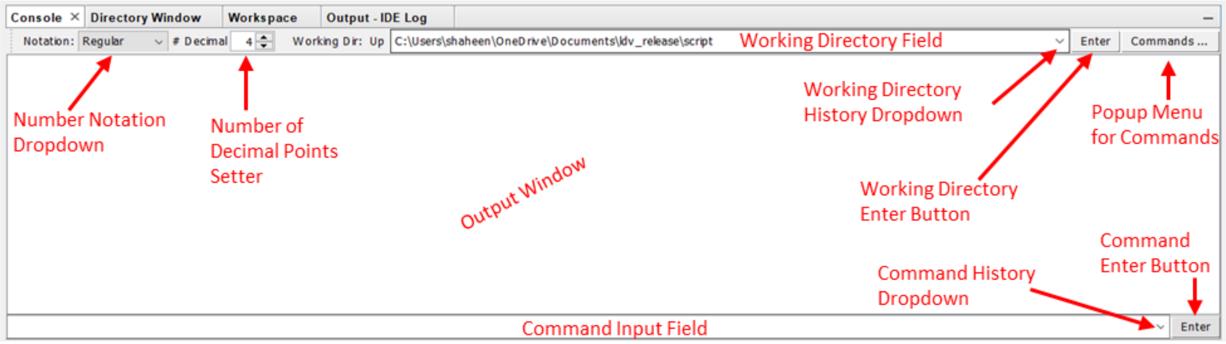
A modified active document can also be saved by using the keyboard shortcut **CTRL+S**. If the active document has already been saved (or opened from an existing document) The **Save** action will update the existing file. If the active document has never been saved, the **Save** action will be similar to the **Save As** action, which is described next.

Selecting the menu item **[Save As]** or using the keyboard shortcut **CTRL+Shift+S** will open a dialog box as shown in Section 4.7.

The dialog box is used to navigate to the location where the file is to be saved. A name of the file must be typed in the **File Name** text field. Clicking on the **[Save]** button will save the file with the name provided in the text field. Clicking on the **[Cancel]** button will save the **Save** operation.

6.2 Using Console and Command History

Console and Command History windows are generally used together. Console is used to directly interact with the system. The command window is presented here again for convenience.



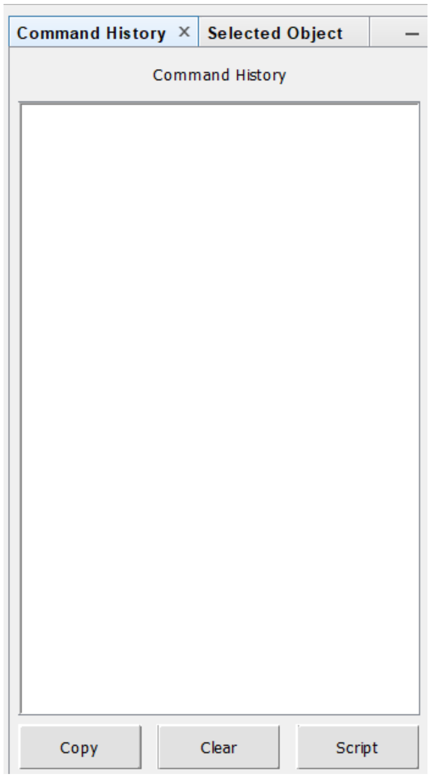
Notation: Regular

# Decimal 4

Working Dir: C:\Users\shaheen\OneDrive\Documents\ldv_release\script

A command can be inputted directly in the Command Input Field. A command can be a Hyper command, a Hyper statement, or a Hyper script file name. Details of the Hyper commands, statements, and scripts can be found in the Hyper Reference Manual. After typing the command, either clicking on the Command Enter button or pressing the Enter key on the keyboard executes the command. Outputs are displayed in the Output window. The Command History Dropdown displays a limited number of previous commands. Selecting any item from the dropdown execute the corresponding command.

A longer list of command history is also maintained in the Command History panel of the Property pane. The Command History panel is presented here again for convenience.

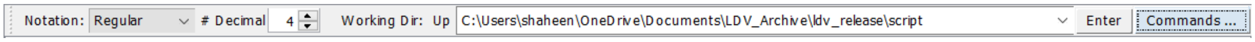


Individual commands appear as single items in the middle section. Commands can be individually selected one or multiple command can be selected by holding down the **CTRL** key and clicking on the chosen commands. All the command in the Command History can be selected by using **CTRL+A** key combination.

The selected commands can be copied to the clipboard by clicking the **Copy** button. The copied commands can be pasted in the Command Input Field of the console or in a script editor. Alternately, the selected commands can be inserted in a new script by clicking on the **Script** button. After adding commands to a script editor, the script can be further modified.

The Command History window can be cleared by clicking on the **Clear** button.

The toolbar above the Output window is depicted below:



Notation: Regular

Decimal 4

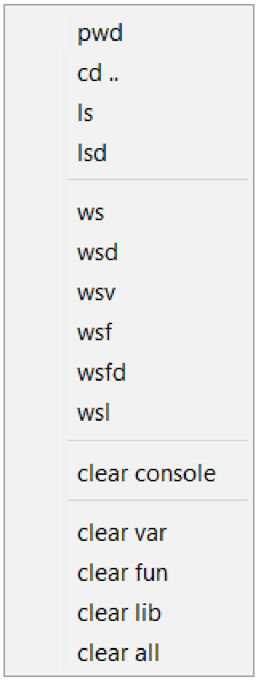
Working Dir: Up C:\Users\shaheen\OneDrive\Documents\LDV_Archive\ldv_release\script

The toolbar contains some utility tools. The Notation option is used to specify the notation in which results will be displayed. The options are regular, scientific, and engineering.

# Decimal option is used to specify the number of decimal places that the result is displayed in.

Working directory is the directory from which scripts are run. The working directory can be changed using the change directory command or function. Refer to the Hyper Reference Manual for detail.

Clicking on “Commands…” button displays a popup menu that contains a list of commonly used commands that can be used conveniently without having to type them in manually. The popup menu is depicted below:



A detailed description of all these options given in the table below:

Command	Stands for	Description
`pwd`	Print Working Directory	Selecting this menu item or typing "**`pwd`**" in the Command Input Field prints out the current working directory path.
`cd ..`	Change Directory to the parent directory	Selecting this menu item or typing "**`cd ..`**" in the Command Input Field causes the current working directory to be set to the directory that is one level above the current working.
`ls`	List	Selecting this menu item or typing "**`ls`**" in the Command Input Field prints the contents of the current working directory in the Output Window.
`lsd`	List in detail	Selecting this menu item or typing "**`lsd`**" in the Command Input Field prints the contents of the current working directory in the Output Window in detail format.
`ws`	Workspace	Selecting this menu item or typing "**`ws`**" in the Command Input Field prints the **contents** of the current workspace in the Output Window.
`wsd`	Workspace in detail	Selecting this menu item or typing "**`wsd`**" in the Command Input Field prints the **contents** of the current workspace in the Output Window in **detail** format.
`wsv`	Workspace with values	Selecting this menu item or typing "**`wsv`**" in the Command Input Field prints the **contents** with their **values** of the current workspace in the Output Window in detail format.
`wsf`	Workspace functions	Selecting this menu item or typing "**`wsf`**" in the Command Input Field prints the only the **function** and their parameters that are in the current workspace in the Output Window.
`wsfd`	Workspace function in detail	Selecting this menu item or typing "**`wsfd`**" in the Command Input Field prints the only the **function** and their parameters that are in the current workspace in the Output Window in **detail** format.
`wsl`	Workspace libraries	Selecting this menu item or typing "**`wsl`**" in the Command Input Field prints the only the **libraries** and their parameters that are in the current workspace in the Output.
`clear console`	Clear console	Selecting this menu item or typing "**`clear console`**" in the Command Input Field clears **output messages** from the Output Window of the console.

`clear var`	Clear variables	Selecting this menu item or typing “**`clear var`**” in the Command Input Field clears only the **variables** from the current workspace.
`clear fun`	Clear functions	Selecting this menu item or typing “**`clear fun`**” in the Command Input Field clears only the **functions** from the current workspace.
`clear lib`	Clear libraries	Selecting this menu item or typing “**`clear lib`**” in the Command Input Field clears only the **libraries** from the current workspace.
`clear all`	Clear everything	Selecting this menu item or typing “**`clear fun`**” in the Command Input Field clears **everything** from the current workspace.

The Workspace app can be used to interact with the workspace graphically.

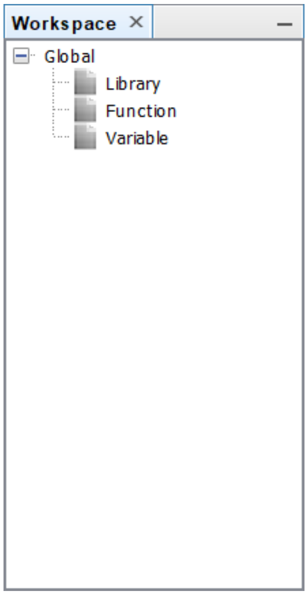
6.3 Using Apps

There are two different kinds of apps in LDV: Apps that produce components for documents and apps that do not. Apps that do not produce components for documents can be thought of as utility apps related to the LDV system, such as the Workspace and File Explorer apps. These apps are called Utility Apps. Apps that produce components for documents are called Document Component Apps. Currently, there are four Document Component Apps: Text app, Table app, Script app, and Plot app. Some of these app’s functions can be accessed using TUI.

Utility Apps

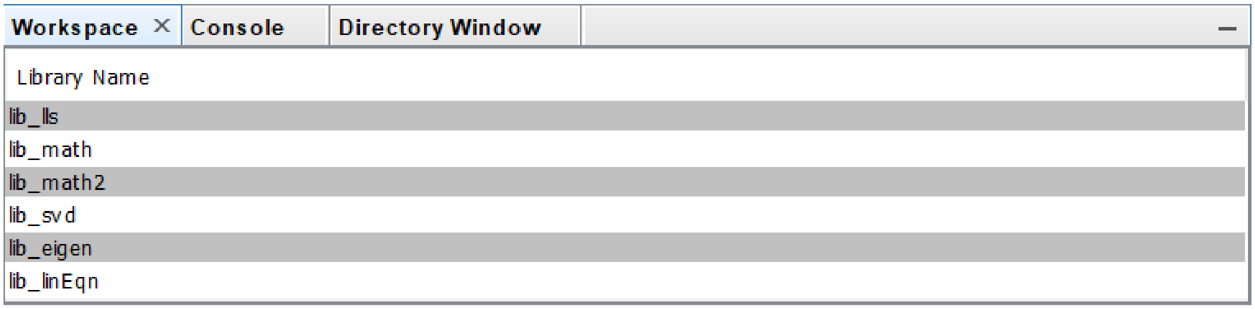
6.3.1 Using Workspace App

As seen in the previous section, the user can interact with workspace by issuing commands from the console. The user can also interact with the workspace using graphical tool, Workspace app. The Workspace app GUI consists of a tool panel in the App pane, a tab in the System pane, and a Property panel in the Selected Object tab of the Property pane. The Workspace tool pane is depicted below:



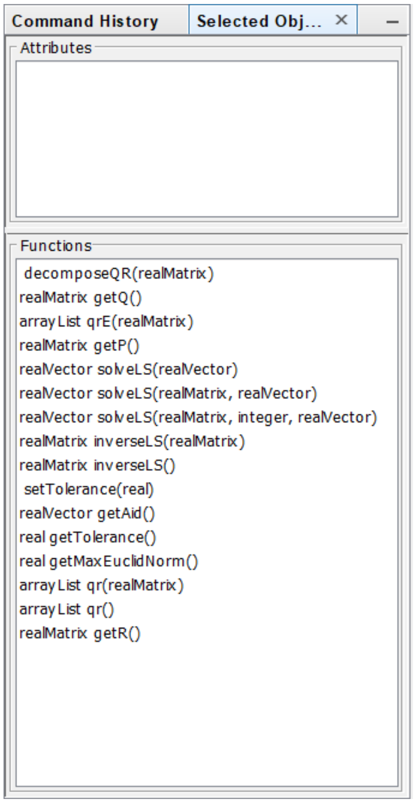
The workspace is organized hierarchically by a tree. The contents of the workspace are grouped by libraries, functions, and variables. Selecting any item on the tree activates the Workspace tab of the System pane and displays the content of the selected item.

The Workspace tab with the Library item selected from the tree in the Workspace tool pane is depicted below:



Library Name
lib_lls
lib_math
lib_math2
lib_svd
lib_eigen
lib_linEqn

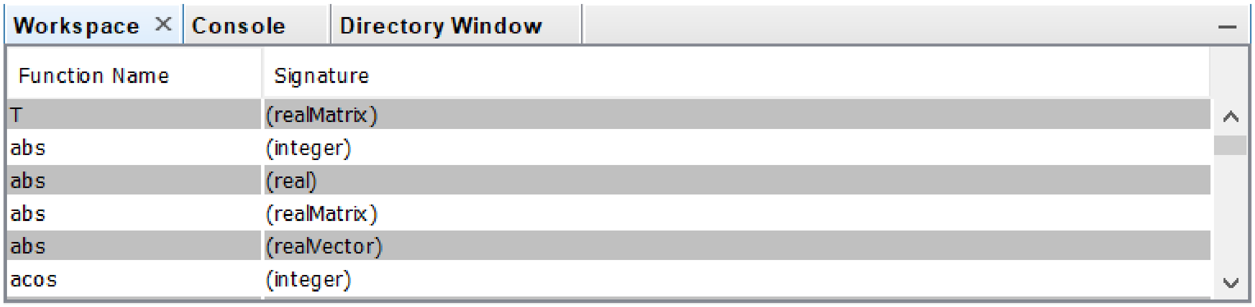
The figure above shows a list of libraries in the current workspace. Selecting a library item from the Workspace window activates the property panel of the selected library. An example of the Library property panel is depicted below:



```
 decomposeQR(realMatrix)
realMatrix getQ()
arrayList qrE(realMatrix)
realMatrix getP()
realVector solveLS(realVector)
realVector solveLS(realMatrix, realVector)
realVector solveLS(realMatrix, integer, realVector)
realMatrix inverseLS(realMatrix)
realMatrix inverseLS()
 setTolerance(real)
realVector getAid()
real getTolerance()
real getMaxEuclidNorm()
arrayList qr(realMatrix)
arrayList qr()
realMatrix getR()
```

The figure above shows the functions with their input and output parameters in the selected library. The example library does not have any attribute. If it had, they would be displayed in the Attributes box. Attributes are constants associated with the library such as PI.

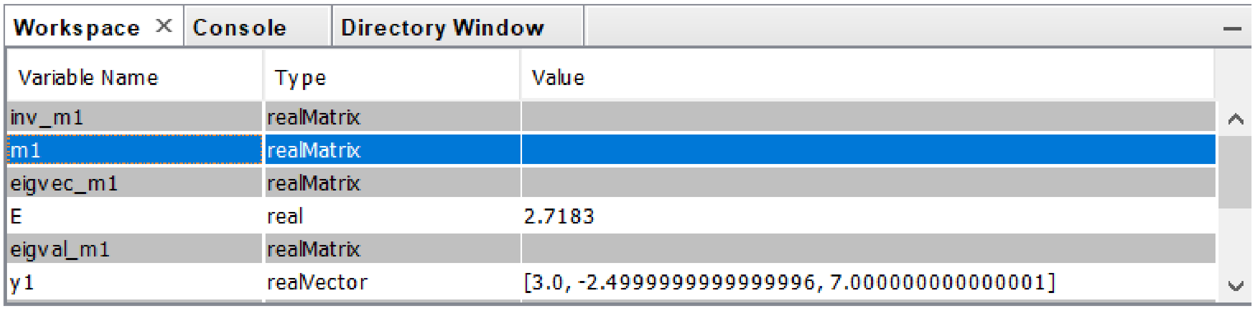
The Workspace tab with the Function item selected from the tree in the Workspace tool pane is depicted below:



Function Name	Signature
T	(realMatrix)
abs	(integer)
abs	(real)
abs	(realMatrix)
abs	(realVector)
acos	(integer)

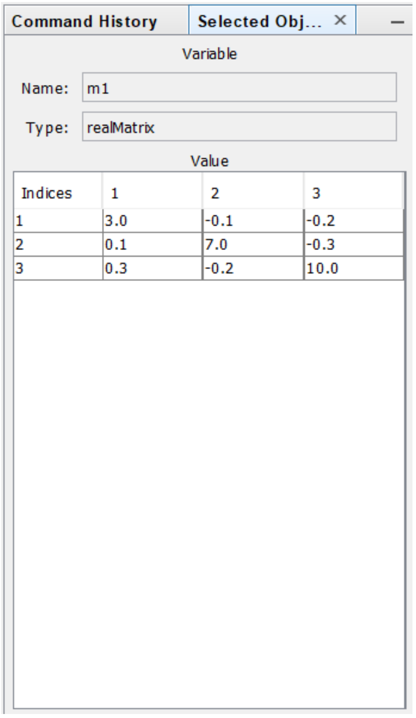
There are no property panel associated with the functions.

The Workspace tab with the Variable item selected from the tree in the Workspace tool pane is depicted below:



Variable Name	Type	Value
inv_m1	realMatrix	
m1	realMatrix	
eigvec_m1	realMatrix	
E	real	2.7183
eigval_m1	realMatrix	
y1	realVector	[3.0, -2.4999999999999996, 7.000000000000001]

The figure above shows a table of variables in the current workspace. The first column contains the variable names. The second column contains the types of the variables. The third column contains the values for some variables. If the value of a variable contains too many components, such as the case for the variable **`m1`** in the above figure, its value is not displayed in the table. Instead, the value is displayed in the property panel of the variable. Selecting a variable item from the Workspace window activates the property panel of the selected library. An example of the Variable property panel is depicted below:



Variable

Name: m1

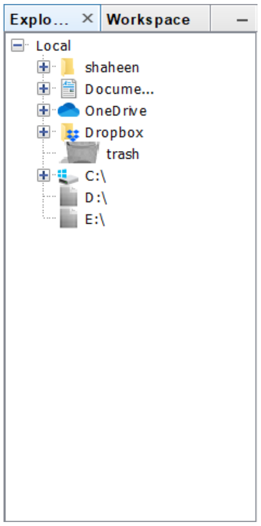
Type: realMatrix

Value

Indices	1	2	3
1	3.0	-0.1	-0.2
2	0.1	7.0	-0.3
3	0.3	-0.2	10.0

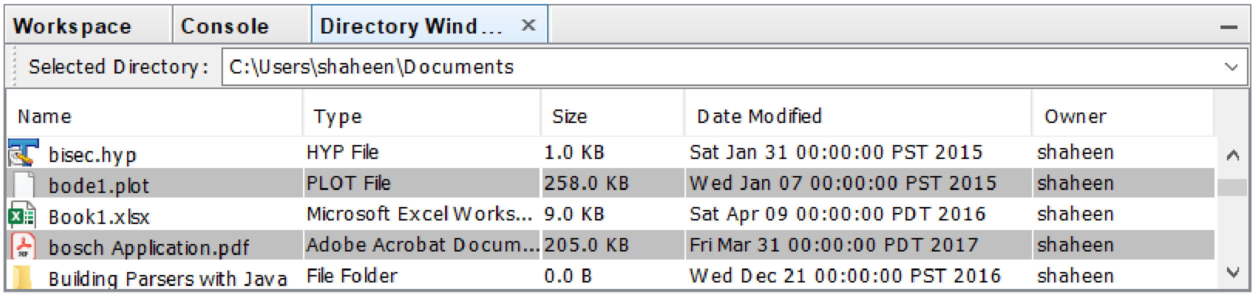
6.3.2 Using File Explorer App

The File Explorer app GUI consists of a tool panel in the App pane, a tab in the System pane, and a Property panel in the Selected Object tab of the Property pane. The File Explorer tool pane is depicted below:



The figure above shows the computer system file directory tree. The directories can be traversed using the tree. The tree works similar to the file explorers in the popular operating systems, such as Windows® and MacOS®. Selecting any item on the tree activates the Directory tab of the System pane and displays the content of the selected directory.

The Directory tab with the selected directory is depicted below:

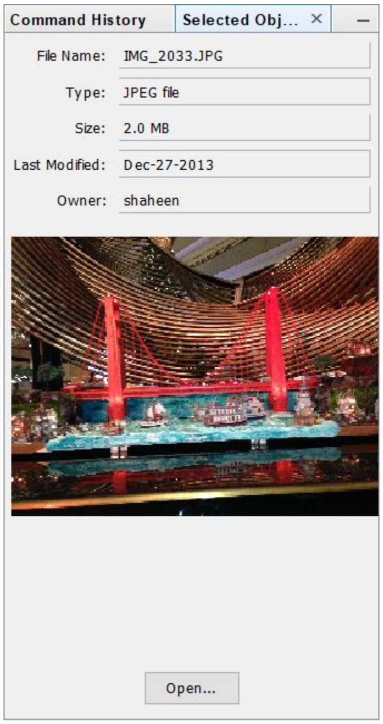


Selected Directory: C:\Users\shaheen\Documents

Name	Type	Size	Date Modified	Owner
bisec.hyp	HYP File	1.0 KB	Sat Jan 31 00:00:00 PST 2015	shaheen
bode1.plot	PLOT File	258.0 KB	Wed Jan 07 00:00:00 PST 2015	shaheen
Book1.xlsx	Microsoft Excel Works...	9.0 KB	Sat Apr 09 00:00:00 PDT 2016	shaheen
bosch Application.pdf	Adobe Acrobat Docum...	205.0 KB	Fri Mar 31 00:00:00 PDT 2017	shaheen
Building Parsers with Java	File Folder	0.0 B	Wed Dec 21 00:00:00 PST 2016	shaheen

The figure above shows a table of files in the selected directory. The first column contains the file names. The second column contains the types of the files. The third column contains the size of the files. The fourth column contains the dates the files were modified. And the fifth column contains the names of the owners of the files. The toolbar on the top of the directory window shows the path of the selected directory.

Selecting a file from the Directory window activates the property panel of the selected file. The property panel of a file is customized for each file type. Generally, the properties include the file name, file type, file size, date of the last modification, and the file owner’s name. In addition, either the icon for the file or a preview of the file is displayed. An example of the File property panel of a picture file is depicted below:



File Name: IMG_2033.JPG

Type: JPEG file

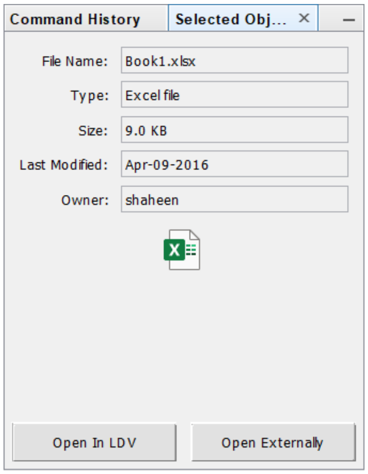
Size: 2.0 MB

Last Modified: Dec-27-2013

Owner: shaheen

This Property panel shows the preview of the file. Clicking on the **[Open …]** button will open the file with the default photo application.

An example of the File property panel of an Excel is depicted below:



File Name: Book1.xlsx

Type: Excel file

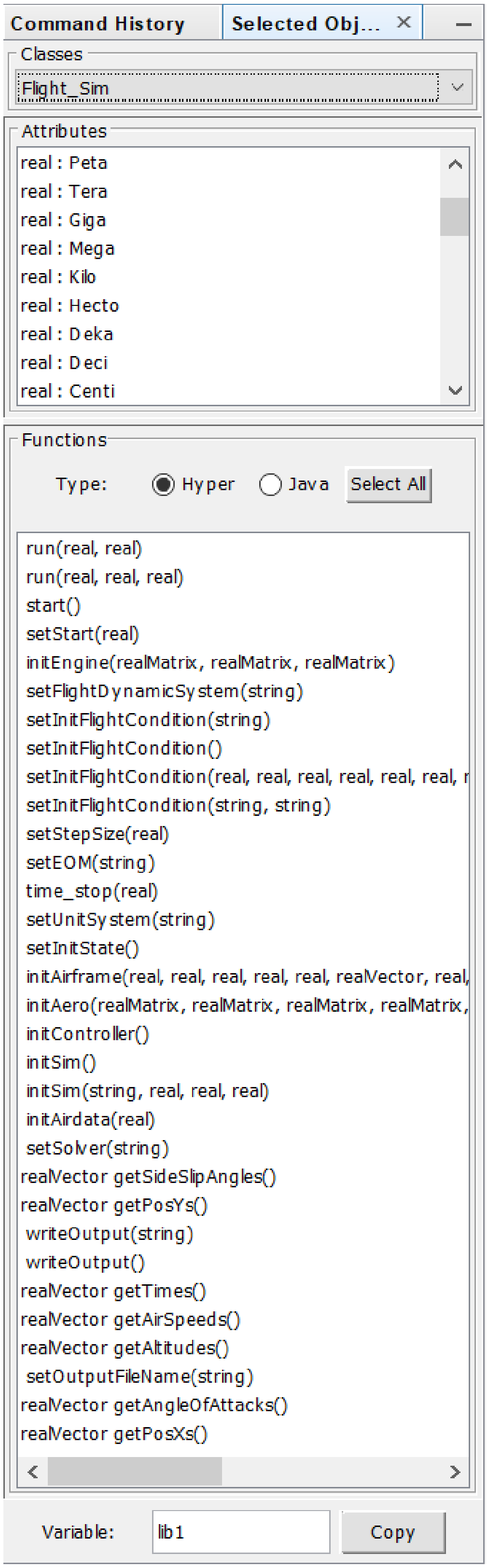
Size: 9.0 KB

Last Modified: Apr-09-2016

Owner: shaheen

This property panel shows an icon of the file instead of preview. At the bottom of the panel, there are two buttons: **[Open In LDV]** and **[Open Externally]**. Clicking on **[Open In LDV]** button imports the file inside the LDV and allows LDV tools to manipulate it. Clicking on **[Open Externally]** button opens the file externally in the file's native application if the native application is available. If the file cannot be opened in the LDV or the file's native application is not available, the corresponding button will be disabled.

The next example shows the property panel of a JAR file.



Classes

Flight_Sim

Attributes

```
real : Peta
real : Tera
real : Giga
real : Mega
real : Kilo
real : Hecto
real : Deka
real : Deci
real : Centi
```

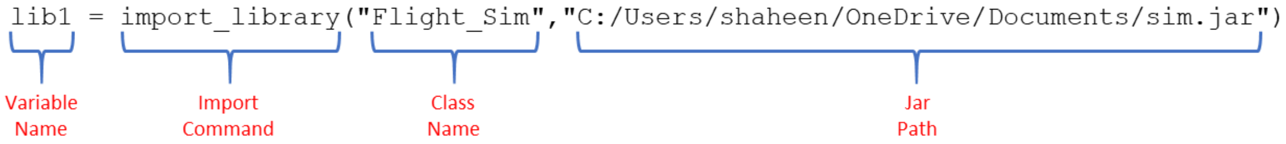
Functions

Type: ● Hyper ○ Java

```
run(real, real)
run(real, real, real)
start()
setStart(real)
initEngine(realMatrix, realMatrix, realMatrix)
setFlightDynamicSystem(string)
setInitFlightCondition(string)
setInitFlightCondition()
setInitFlightCondition(real, real, real, real, real, real, r
setInitFlightCondition(string, string)
setStepSize(real)
setEOM(string)
time_stop(real)
setUnitSystem(string)
setInitState()
initAirframe(real, real, real, real, real, realVector, real,
initAero(realMatrix, realMatrix, realMatrix, realMatrix,
initController()
initSim()
initSim(string, real, real, real)
initAirdata(real)
setSolver(string)
realVector getSideSlipAngles()
realVector getPosYs()
writeOutput(string)
writeOutput()
realVector getTimes()
realVector getAirSpeeds()
realVector getAltitudes()
setOutputFileName(string)
realVector getAngleOfAttacks()
realVector getPosXs()
```

Variable: lib1

There are four parts to this property panel. The top part contains a dropdown that allows to select a class in the JAR file. Note that only the classes from the default package will be displayed in the dropdown. After a class is selected, the attributes of the selected class are displayed in the **Attributes** box and the functions (methods) of the selected class are displayed in the **Functions** box. At the bottom of the panel there is a text field and a button **[Copy]**. The text field is prefilled with a generic variable name. The variable name in the text field can be edited. If the **[Copy]** button is clicked, an import statement for the selected class is copied into the clipboard. Here is an example of an import statement:



```
lib1 = import_library("Flight_Sim","C:/Users/shaheen/OneDrive/Documents/sim.jar")
```

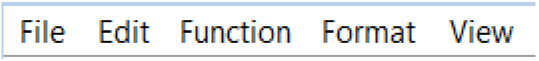
The import statement contains an import command, a class name, and a jar path. A pointer to the imported class is assigned to a variable of the specified name. The statement can be pasted in a script for later use or in the Command Input Field for immediate execution. The import statement can also be executed by pressing the **[ENTER]** key in the keyboard when the variable text field has the focus.

Document Component Apps

All the Document Component Apps share common menus, menu items, and toolbar tools. In addition, some Document Component Apps have menu items and toolbar tools that are specific to the particular app. In the next two subsections, the common menu items and common toolbar tools are described.

Common App Menu Bar

A common app menu bar is depicted below:

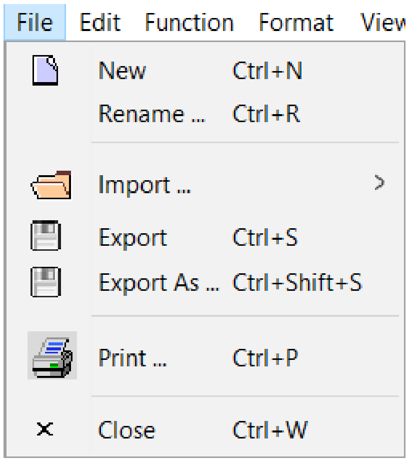


Menus that are common to all apps are:

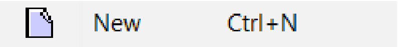
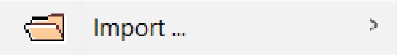
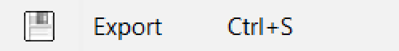
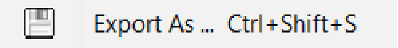
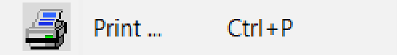
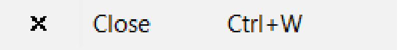
- File
- Edit
- Functions
- Format
- View

File Menu:

A common File menu of an app menu bar is depicted below:

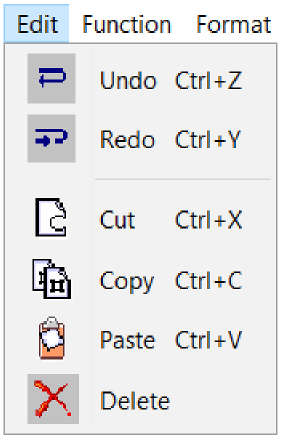


The common File menu items are described in the table below:

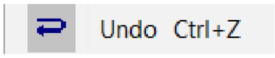
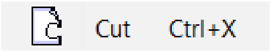
Menu Item	Description
 New Ctrl+N	Create a new content in the app frame. This action removes the existing content. The keyboard short cut is **CRTL+N**.
Rename ... Ctrl+R	Renames the frame. The keyboard short cut is **CRTL+R**.
 Import ... >	Imports an existing content. A dialog is opened when this menu item is selected. The dialog box is used to select a file to be imported. Import is similar to Open. The name Import is used to distinguish it from the Open menu item in the Main Menu Bar.
 Export Ctrl+S	Exports the contents of the app frame to a different file. A dialog is opened when the menu item Export is selected. The dialog box is used to create a file in which the frame content will be exported. Export is similar to Save. The name Export is used to distinguish it from the Save menu item in the Main Menu Bar. The keyboard short cut is **CRTL+S**.
 Export As ... Ctrl+Shift+S	Exports the frame content to a different file. The keyboard short cut is **CRTL+Shift+S**.
 Print ... Ctrl+P	Prints the content of the frame. A dialog is opened when this menu item is selected. Print preferences can be selected in the dialog box. The keyboard short cut is **CRTL+P**.
 × Close Ctrl+W	Closes the app frame. The keyboard short cut is **CRTL+W**.

Edit Menu:

An Edit menu of an app menu bar is depicted below:



The common Edit menu items are described in the table below:

Menu Item	Description
 Undo Ctrl+Z	Reverses the previous action. The keyboard short cut is **CRTL+Z**.
Redo Ctrl+Y	Repeats the previous action. The keyboard short cut is **CRTL+Y**.
 Cut Ctrl+X	Copies the selected item into the clipboard and deletes the selected item. The keyboard short cut is **CRTL+X**.
Copy Ctrl+C	Copies the selected item, but it does not delete the selected item. The keyboard short cut is **CRTL+C**.
Paste Ctrl+V	Copies the item from the clipboard to the content of the app frame. The keyboard short cut is **CRTL+V**.
Delete	Removes the selected item.

Not every item from this menu is currently available for all the apps.

Functions Menu:

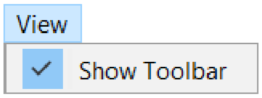
A Function menu contains functions that are specific to an app; therefore, there are no common menu items.

Format Menu:

A Format menu contains functions that are specific to an app; therefore, there are no common menu items.

View Menu:

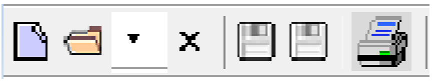
A common View menu of an app menu bar is depicted below:



Show Toolbar Item: Selecting this menu item makes the app toolbar visible. The app toolbar is hidden by default.

Common App Toolbar

The toolbar on the app frame is invisible by default. It can be made visible from the View menu by checking View > Toolbar. A common app toolbar is depicted below:



In addition to the common toolbar, some apps have additional toolbars specific to the app. Descriptions of the buttons are presented in the table below:

Icon	Name	Description
🗋	New	This button is clicked to create a new content in the app frame. This action removes the existing content.
📂	Import	This button is clicked to import an existing content. Clicking this button opens a dialog box. The dialog box is used to select a file to be imported.
▾	File History	Clicking on this button displays a list of previously imported files. An item from the list can be selected to import the corresponding file.
✕	Delete	Clicking on this button clears the content from the app frame.
💾	Export	Clicking on this button exports the content of the app frame.
💾	Export As	This button is used to export the contents of the app frame to a different file. Clicking on this button open a dialog box. The dialog box is used to create a file in which the frame content will be exported.
🖨	Print	This button is used to print the content of the app frame.

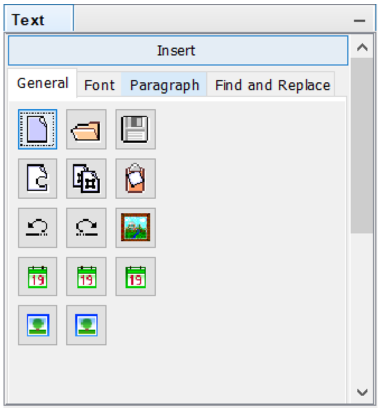
6.3.3 Using Text App

The Text app is used to add formatted text to a document. The app provides a comprehensive set of text formatting tools. The app creates components that are similar to word processor documents.

Tool Panel

The tool panel for this app is organized into several tabs: **[General]**, **[Font]**, **[Paragraph]**, and **[Find and Replace]**. Each tab contains several related text formatting tools. A button **[Insert]** is available regardless of which tab is selected. This button is a toggle button, and it is selected to insert a Text component in a document. The tabs are presented below:

General Tab:

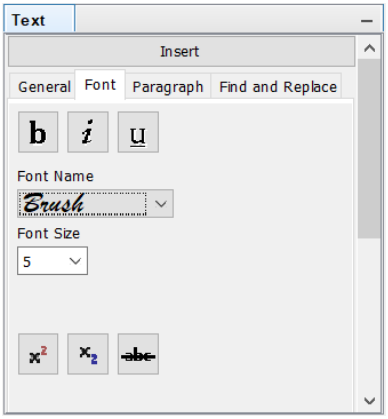


The figure above shows the **General** tab. It contains 14 buttons. Descriptions of the button are given in the table below:

Icon	Name	Description
🗋	New	Selected to insert a new Text component in a document.
📂	Import	Selected to import a text from a file.
💾	Export	Exports the content of the active text component.
✂	Cut	Copies selected text from the active text component into the clipboard and delete the selected text.
⧉	Copy	Copies selected text from the active text component into the clipboard but does not delete the selected text.
📋	Paste	Pastes the content of the clipboard into the cursor location of the active text component.
↶	Undo	Reverses the last action.

↷	Redo	Repeats the last action.
🖼	Insert Image	Inserts image in the cursor location of the active text component.
19	Insert Date MM/DD/YYYY	Inserts the current date in the cursor location of the active text component in MM/DD/YYYY format.
19	Insert Date MMM DD, YYYY	Inserts the current date in the cursor location of the active text component in MMM DD, YYYY format.
19	Insert Date MMMM DD, YYYY	Inserts the current date in the cursor location of the active text component in MMMM DD, YYYY format.
🖼	Insert Symbol	Inserts symbol in the cursor location of the active text component.

Font Tab:



Font Name: Brush

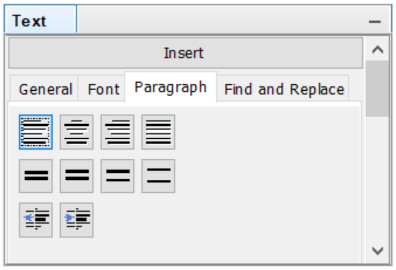
Font Size: 5

The figure above shows the Font tab. Descriptions of the buttons and dropdowns are given in the table below:

Icon	Name	Description
b	Bold	Makes the selected text **bold**.
i	Italic	Makes the selected text *italic*.
u	Underline	Underlines the selected text.
~~abc~~	Strike Through	~~Strikes through~~ the selected text.
x^2	Superscript	Makes the selected text superscript.
x_2	Subscript	Makes the selected text subscript.

Brush	Font Name	Used to select font.
5	Font Size	Used to select font size.

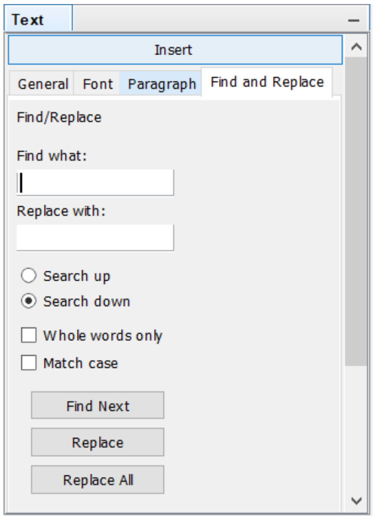
Font Tab:



The figure above shows the **Paragraph** tab. Descriptions of the button are given in the table below:

Icon	Name	Description
≡	Align Left	Aligns the selected paragraph to the left.
≡	Align Center	Aligns the selected paragraph to the center.
≡	Align Right	Aligns the selected paragraph to the right.
≡	Align Justify	Aligns the selected paragraph to justify.
═	Single Space	Puts single spaces between the lines of the selected paragraph.
═	1.5 Space	Puts 1.5 spaces between the lines of the selected paragraph.
═	Double Space	Puts double spaces between the lines of the selected paragraph.
═	Triple Space	Puts triple spaces between the lines of the selected paragraph.
⇤	Decrease Indent	Decreases the indentation of the selected paragraph.
⇥	Increase Indent	Increases the indentation of the selected paragraph.

Find and Replace Tab:



The figure above shows the **Find and Replace** tab. Descriptions of the tools are given in the table below:

Icon	Name	Description
Find what:	Find What	Text field to enter the **search** string.
Replace with:	Replace With	Text field to enter the **replacement** string.
○ Search up	Search Up	Sets the direction of the search to **up**.
● Search down	Search Down	Sets the direction of the search to **down**.
□ Whole words only	Whole World Only	Searches for only whole words.
□ Match case	Match Case	Searches only for strings that match the cases of the search string.
Find Next	Find Next	Find the next match.
Replace	Replace	Replaces the found string.
Replace All	Replace All	Replaces all the matched strings.

Menu Bar

The Text app uses the standard app menu bar and menu items.

Toolbar

The app also uses the standard app toolbar.

Inserting new Text component in a document

A new Text component can be inserted in a document using GUI or command. The following steps will insert a Text component using GUI:

1. Select the **[Insert]** button from the Text tool pane.
2. Either click or click and drag in the selected open document.

If just clicked, a Text app frame of a predetermined size will be created and embedded in the document. If clicked and dragged, the app frame size will be determined by the drag action.

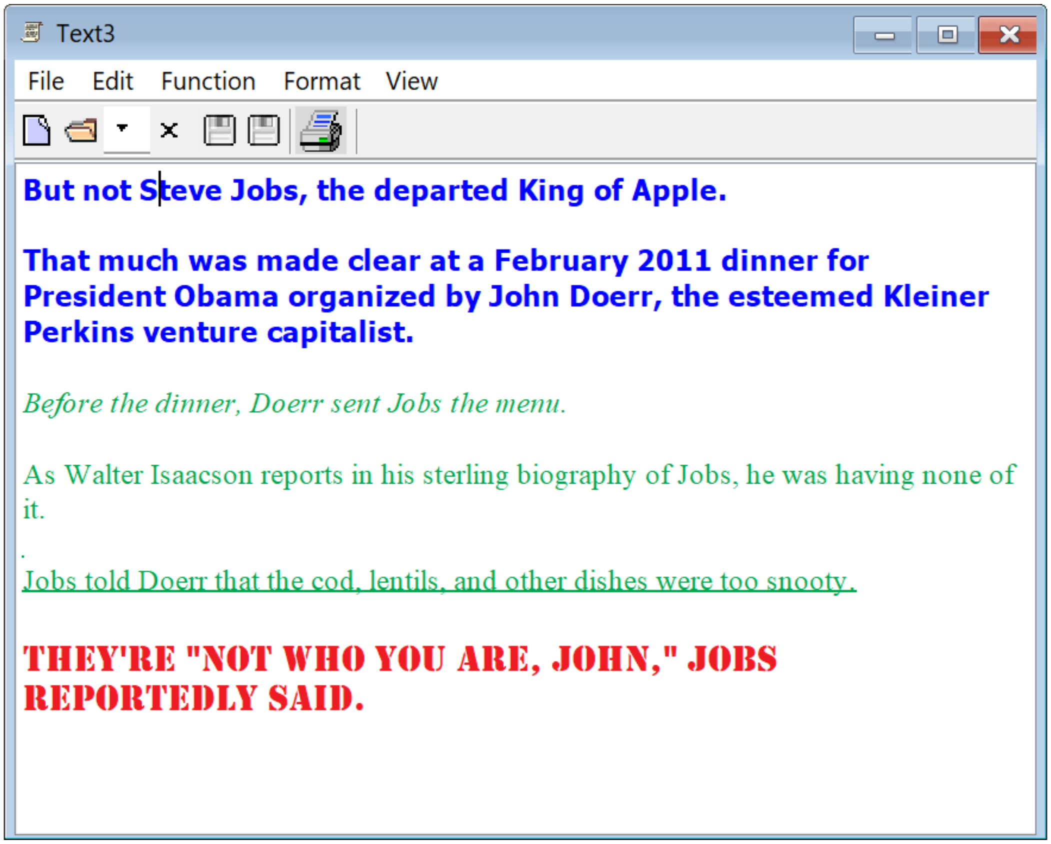
A Text component can also be inserted using the following command:

```
createText(<text component name>)
```

The string inside the angle bracket (< >) is the name the user wants to assign to the inserted component.

Once a text component is inserted, new text can be added and formatted, or text can be imported from a file.

An example of a Text component depicted below:



But not Steve Jobs, the departed King of Apple.

That much was made clear at a February 2011 dinner for President Obama organized by John Doerr, the esteemed Kleiner Perkins venture capitalist.

Before the dinner, Doerr sent Jobs the menu.

As Walter Isaacson reports in his sterling biography of Jobs, he was having none of it.

.

Jobs told Doerr that the cod, lentils, and other dishes were too snooty.

THEY'RE "NOT WHO YOU ARE, JOHN," JOBS REPORTEDLY SAID.

The figure above shows a Text component with the app frame activated. The Content shows formatted text with different font types, size, and colors.

Editing

Editing text is similar to editing in any word processor.

Text App's Text User Interface

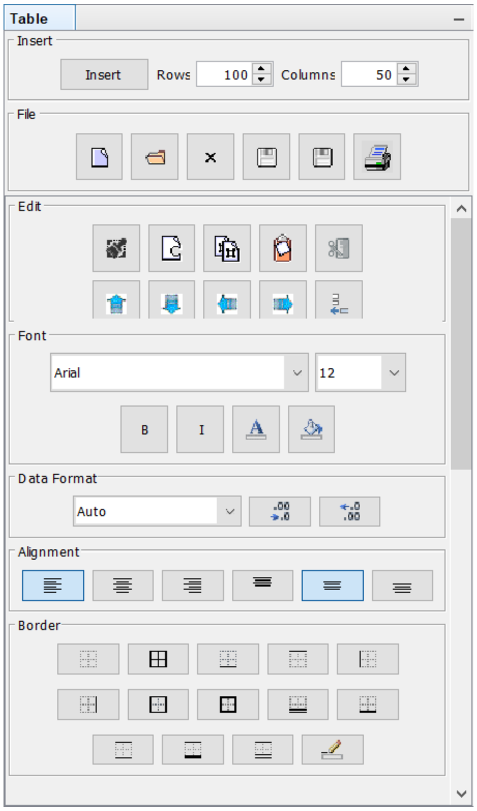
Function	Description
`createText("<text object Name">)`	Creates a new Text object in the selected document. The name of the object is provided by the user in the angle bracket (<>).

6.3.4 Using Table App

The Table app is used to add tabular data to a document. The app provides a comprehensive set of tools for manipulating tabular data. The app creates components that are similar to spread sheet documents.

Tool Panel

The tool panel for this app is shown below:



Rows 100

Columns 50

Font: Arial, 12

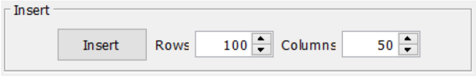
Data Format: Auto

The Table tool panel is organized in several groups:

1. Insert
2. File
3. Edit
4. Font
5. Data format
6. Alignment
7. Border

Insert Group

The Insert group contains an “insert” button and a couple of spinners to enter numbers as shown below:



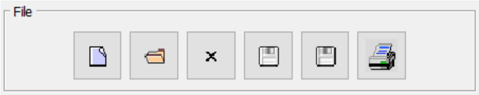
The spinners labeled “Rows” and “Columns” are used to specify the number of rows and columns, respectively, will be in the newly inserted table.

The items in the Insert group are described in the table below:

Icon	Name	Description
Insert	Insert	The “Insert” button is selected to create a new table and insert it in a document.
Rows 100	Rows	Used to specify the number of rows will be in the newly inserted table.
Columns 50	Columns	Used to specify the number of columns will be in the newly inserted table.

File Group

The File group contains tools related to file utilities and is depicted below:



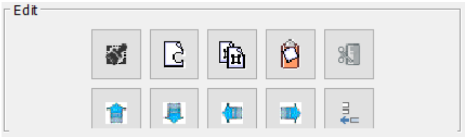
The items in the File group are described in the table below:

Icon	Name	Description

	New	Creates a new table in the selected Table frame. The contents of the existing table is lost.
	Import	Imports an existing table or spread sheet from a file.
✕	Close	Closes the selected table.
	Export	Exports the selected to a file.
	Save As	Exports the selected table in a different name and/or format.
	Print	Prints the selected table.

Edit Group

The Edit group contains tools for editing and is depicted below:



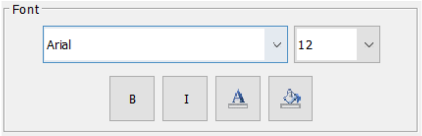
The items of the Edit groups are described in the table below:

Icon	Name	Description
⬆	Fill Up	Fills the cells **above** the selected cell with the copies of the content of the selected cell.
⬇	Fill Down	Fills the cells **below** the selected cell with the copies of the content of the selected cell.
⬅	Fill Left	Fills the cells **left** of the selected cell with the copies of the content of the selected cell.
➡	Fill Right	Fills the cells **right** of the selected cell with the copies of the content of the selected cell.
	Insert Rows	Inserts a new row below the selected row.
	Append Row	Adds a new row at the end of the table.
	Remove Rows	Removes the selected rows.
	Insert Columns	Inserts a new column at the right of the selected column.
	Append Column	Adds a new column at the end of the table.
	Remove Column	Removes the selected columns.
	Clear Table	Clears the entire table.
	Clear Selected	Clears the selected cell of the table.

	Trim Table	Removes the rows and columns after the last non empty row and column.

Font Group

The Font group contains tools for formatting the contents of the selected table and is depicted below:

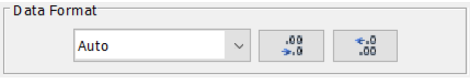


The items of the Font groups are described in the table below:

Icon	Name	Description
Arial	Font Family	Sets the font family for the contents of the selected cells.
12	Fill Right	Sets the font size for the contents of the selected cells.
B	Bold	Sets the font style for the contents of the selected cells to bold.
I	Italic	Sets the font style for the contents of the selected cells to Italic.
A	Text Color	Opens a color chooser dialog box to set the text color for the contents of the selected cells. The Color chooser dialog box is described in Section 4.7
	Background Color	Opens a color chooser dialog box to set the background color of the selected cells. The Color chooser dialog box is described in Section 4.7

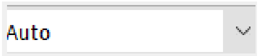
Data Format Group

The Edit group contains tools to format data and is depicted below:

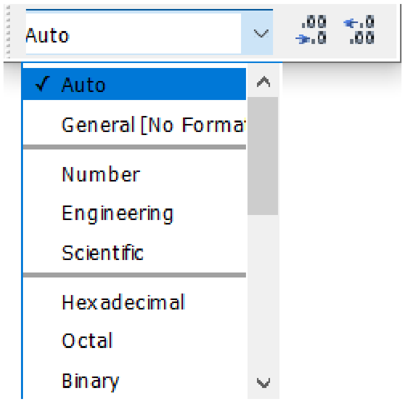


The items of the Data Format group are described in the table below:

Icon	Name	Description

 Auto	Data Format Selector	Dropdown to select the format for the data.
.00 →.0	Decimal Point Decrease	Moves the decimal point to the right.
←.0 .00	Decimal Point Increase	Moves the decimal point to the left.

The expanded view of the Data Format Selector dropdown menu is shown below:



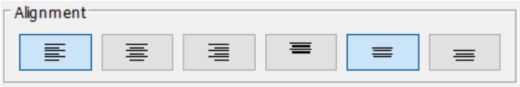
The items in the Data Format selector are described in the table below:

Item	Description	Example
Auto	Default Format.	
General [No Format]	No Format.	
Number	Format as number.	123456789
Engineering	Numbers are displayed in Engineering notation, which is a version of scientific notation in which the exponent of ten must be divisible by three	123.456e9
Scientific	Numbers are displayed in Scientific notation, Nonzero numbers are written in the form $m \times 10^n$ or m times ten raised to the power of n, where n is an integer, and the coefficient m is a nonzero real number (usually between 1 and 10 in absolute value)	123.456e5
Hexadecimal	Numbers are displayed in Hexadecimal (also base 16 or hex) numeral system, which represents numbers using a radix (base) of 16.	
Octal	Numbers are displayed in the Octal numeral system (or oct for short), which is the base-8 number system, and uses the digits 0 to 7.	
Binary	Numbers are displayed in a Binary format, which is expressed in the base-2 numeral system	

Accounting	Numbers are displayed the Accounting format, which contains two decimal points, a thousand separator. The difference between the Accounting format and the Currency format is that the Accounting format puts the dollar sign for example, at the far left end of the cell, and displays zero as a dash.	$ 123.56
Currency	Numbers are displayed the Currency format, which is similar to the Accounting format. However, the decimal points appear aligned in the column and the currency symbol appears next to the first digit.	$123.45
Percent	Displays a percent symbol (%) after the number	89%
Date & Time	Displays the number as date and time	
Date	Displays the number as date	
Time	Displays the number as time	
Boolean	Displays the value in Boolean (True or False)	
Text	Displays the content of the cell as text.	
List	Displays the content of the cell as list.	

Alignment Group

The Alignment group contains tools for aligning the contents of the selected table and is depicted below:

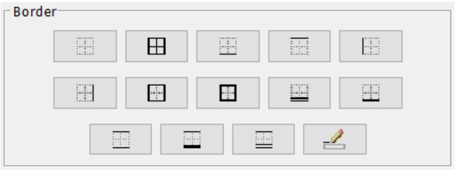


The items of the Font groups are described in the table below:

Icon	Name	Description
≡	Left Alignment	Horizontally aligns the contents of the selected cells to the **left**.
≡	Center Alignment	Horizontally aligns the contents of the selected cells to the **center**.
≡	Right Alignment	Horizontally aligns the contents of the selected cells to the **right**.
≡	Top Alignment	Vertically aligns the contents of the selected cells to the **top**.
≡	Middle Alignment	Vertically aligns the contents of the selected cells to the **middle**.
≡	Bottom Alignment	Vertically aligns the contents of the selected cells to the **bottom**.

Border Group

The Border group contains tools to put borders around selected cells and is depicted below:



The items of the Border group are described in the table below:

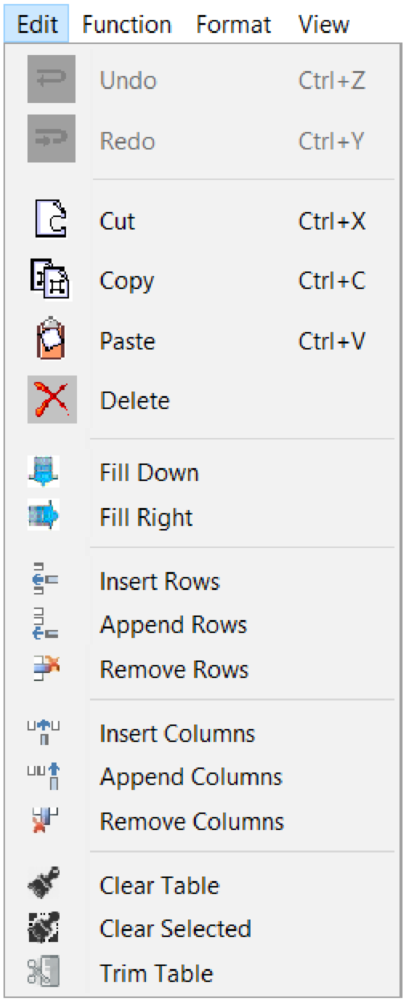
Submenu Item	Description
Bottom Border	Adds border to the bottom of the selected cells.
Top Border	Adds border to the top of the selected cells.
Left Border	Adds border to the left of the selected cells.
Right Border	Adds border to the right of the selected cells.
No Border	Removes all borders.
All Borders	Adds borders to all cells.
Outside Borders	Adds border to the outside of the selected cells.
Thick Box Border	Adds a thick border to the outside of the selected cells.
Bottom Double Border	Adds a double border to the bottom of the selected cells.
Thick Bottom Border	Adds a thick border to the bottom of the selected cells.
Top and Bottom Border	Adds a double border to the top and bottom of the selected cells.
Top and Thick Bottom Border	Adds a thick border to the top and bottom of the selected cells.
Top and Double Bottom Border	Adds a single order to the top and a double border to the bottom of the selected cells.
Line Color	Opens a color selection dialog box. Which can be used to changes the border color. This is the same dialog box as shown in the Font submenu section.

Menu Bar

The Table app uses the standard app menu bar. However, several of the menus have items specific to the Table app. The menus specific to this app are described below:

Edit Menu

The Edit menu for the Table app is shown below:



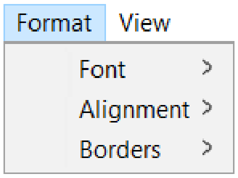
The common Edit menu item were described previously; therefore, only the menu items specific to the Table app are described in the table below:

Menu Item	Description
Fill Down	Fills the cells below the selected cell with the copies of the content of the selected cell.
Fill Right	Fills the cells right of the selected cell with the copies of the content of the selected cell.
Insert Rows	Inserts a new row below the selected row.
Append Rows	Adds a new row at the end of the table.
Remove Rows	Removes the selected rows.
Insert Columns	Inserts a new column at the right of the selected column.
Append Columns	Adds a new column at end of the table.
Remove Columns	Removes the selected columns
Clear Table	Clears the entire table.
Clear Selected	Clears the selected cell of the table.

Trim Table	Removes the rows and columns after the last non empty row and column.

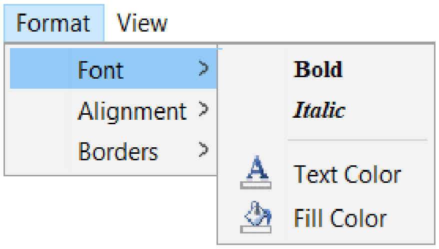
Format Menu

The Format menu for the Table app is shown below:



Font Submenu

The Font submenu is shown below:



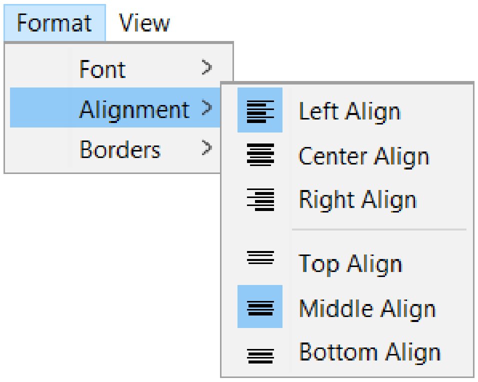
The items of the Font submenu are described in the table below:

Submenu Item	Description
Bold	Makes the selected cell contents bold.
Italic	Makes the selected cell contents italic.
Text Color	Opens a color selection dialog box. Which can be used to changes the text color.
Fill Color	Opens a color selection dialog box. Which can be used to changes the cell background color.

The color selection dialog box is shown in Section 4.7

Alignment Submenu

The Alignment submenu is depicted below:

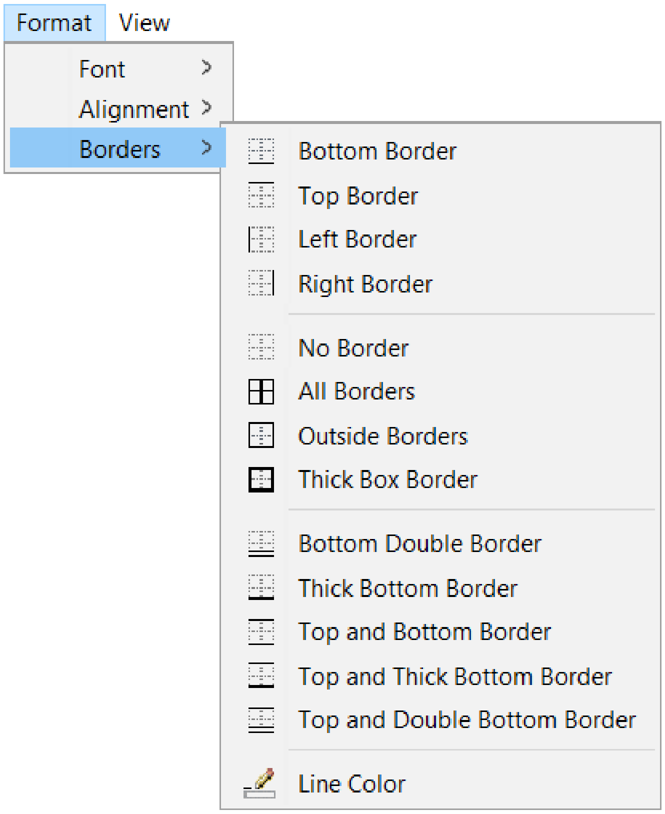


The items of the Alignment submenu are described in the table below:

Submenu Item	Description
Left Align	Horizontally aligns the contents of the selected cells to the **left**.
Center Align	Horizontally aligns the contents of the selected cells to the **center**.
Right Align	Horizontally aligns the contents of the selected cells to the **right**.
Top Align	Vertically aligns the contents of the selected cells to the **top**.
Middle Align	Vertically aligns the contents of the selected cells to the **middle**.
Bottom Align	Vertically aligns the contents of the selected cells to the **bottom**.

Border Submenu

The Border submenu is depicted below:



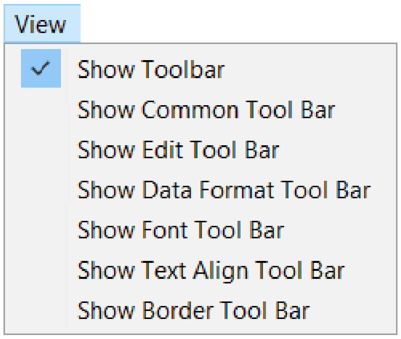
The items of the Border submenu are described in the table below:

Submenu Item	Description
Bottom Border	Adds border to the bottom of the selected cells.
Top Border	Adds border to the top of the selected cells.
Left Border	Adds border to the left of the selected cells.
Right Border	Adds border to the right of the selected cells.
No Border	Removes all borders.
All Borders	Adds borders to all cells.
Outside Borders	Adds border to the outside of the selected cells.
Thick Box Border	Adds a thick border to the outside of the selected cells.
Bottom Double Border	Adds a double border to the bottom of the selected cells.
Thick Bottom Border	Adds a thick border to the bottom of the selected cells.
Top and Bottom Border	Adds a double border to the top and bottom of the selected cells.
Top and Thick Bottom Border	Adds a thick border to the top and bottom of the selected cells.
Top and Double Bottom Border	Adds a single order to the top and a double border to the bottom of the selected cells.

Line Color	Opens a color selection dialog box. Which can be used to changes the border color. This is the same dialog box as shown in the Font submenu section.

View Menu

The View menu for the Table app is shown below:



Only the menu items specific to the Table app are described in the table below:

Menu Item	Description
Show Common Tool Bar	Shows or hides the toolbar that contains the commonly used tools in the Table app.
Show Edit Tool Bar	Shows or hides the toolbar that contains tools related to editing.
Show Data Format Tool Bar	Shows or hides the toolbar that contains tools related to data formatting.
Show Font Tool Bar	Shows or hides the toolbar that contains tools related to changing font characteristics.
Show Text Align Tool Bar	Shows or hides the toolbar that contains tools related to cell content alignment.
Show Border Tool Bar	Shows or hides the toolbar that contains tools related to modifying borders.

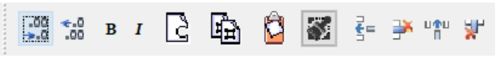
A checkmark before the menu item indicates if a particular toolbar is visible.

Toolbar

The Table app uses several toolbars specific to this app, in addition to the standard app toolbar.

Typical Toolbar

The Typical toolbar contains tools that are typically used. The Typical toolbar is depicted below:

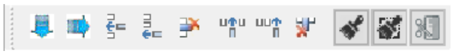


The items of the Typical toolbar are described in the table below:

Icon	Name	Description
	Decimal Point Decrease	Moves the decimal point to the right.
	Decimal Point Increase	Moves the decimal point to the left.
B	Bold	Makes the selected cell contents bold.
I	Italic	Makes the selected cell contents italic.
	Cut	Copy contents of the selected cells into the clipboard and delete the selected text.
	Copy	Copies contents of the selected cells into the clipboard but does not delete the selected text.
	Paste	Pastes the content of the clipboard at the selected cell of the table.
	Clear Selected	Clears the selected cell of the table.
	Insert Rows	Inserts a new row below the selected row.
	Remove Rows	Removes the selected rows.
	Insert Columns	Inserts a new column at the right of the selected column.
	Remove Columns	Removes the selected columns.

Edit Toolbar

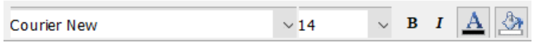
The Edit toolbar is depicted below:



The items of the Edit toolbar are described in the table in the description of the Edit group of the tool panel.

Font Toolbar

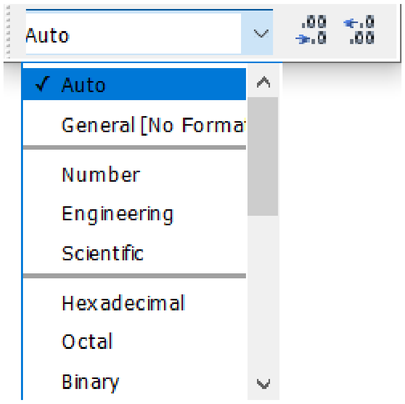
The Font toolbar is depicted below:



The items of the Font toolbar are described in the table in the description of the Font group of the tool panel.

Data Format Toolbar

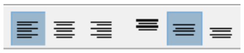
The Edit toolbar is depicted below:



The items of the Data Format toolbar are described in the table in the description of the Data Format group of the tool panel.

Alignment Toolbar

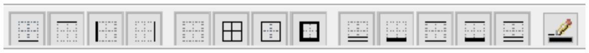
The Alignment toolbar is depicted below:



The items of the Alignment toolbar are described in the table in the description of the Alignment group of the tool panel.

Border Toolbar

The Border toolbar is depicted below:



The items of the Border toolbar are described in the table in the description of the Border group of the tool panel.

Inserting new Table component in a document

A new Table component can be inserted in a document using GUI or command. The following steps will insert a Table component using GUI:

1. Select the **[Insert]** button from the Table tool pane.
2. Either click or click and drag in the selected open document.

If just clicked, a Table app frame of a predetermined size will be created and embedded in the document. If clicked and dragged, the app frame size will be determined by the drag action.

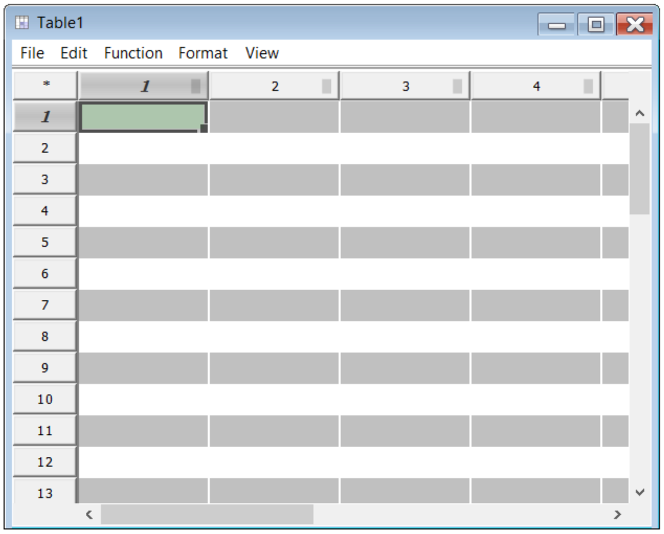
A Table component can also be inserted using the following command:

```
createTable(<table object name>)
```

The string inside the angle bracket (< >) is the name the user wants to assign to the inserted component.

Once a table component is inserted, new table can be added, manipulated, and formatted or table can be imported from a file.

An example of a Table component depicted below:



Editing

Entering Data in a Table

Data can be entered in a table simply by selecting a cell on the table and typing. After finished typing, the key **[Enter]** or the key **[TAB]** on the keyboard must be pressed. Otherwise, the typed data will be lost. If the key **[Enter]** is pressed, the cell below will be selected next. If the key **[TAB]** is pressed, the cell on the right will be selected next.

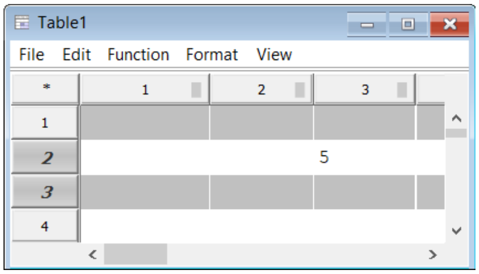
Data can also be entered using the Text User Interface (TUI). The syntax for entering data in a single cell is

```
<Table Name> [<row> , <column>] = data
```

Example:

```
Table1[2,3] = 5
```

After executing this statement, the table look like the figure below:



*	1	2	3
1			
2			5
3			
4			

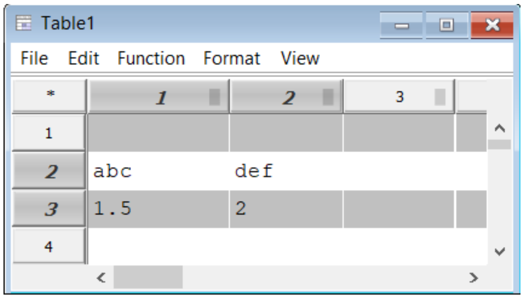
The syntax for entering data in multiple cells is

<Table Name> [<row start : row end> , <column start : column end>] = {data$_1$, data$_2$ … data$_n$}

Example:

```
Table1[2:3,1:2] = {"abc", "def", 1.5, 2}
```

After executing this statement, the table look like the figure below:



*	1	2	3
1			
2	abc	def	
3	1.5	2	
4			

The number of data in the list (inside the curly braces) right of the equal (=) must match the number of data implied by the indices inside the square brackets left of the equal (=). In this case, the number of rows specified is 2 (2 to 3), and the number of columns specified is 2 (1 to 2). So, the total number of data specified by the indices is 4, which is the same as the number of data in the list.

Entering Formulas in a Table

Formulas can be entered in a selected cell by first typing the character equal ('=') then typing the rest of the formula. The syntax for formulas are the same as that of the Hyper (Refer to Hyper Reference Manual for detail). Other cells can be referred in the formula by typing the coordinate row and column numbers of the referred cell inside square brackets, e.g. [<row>, <column>]. Other cells can also be referred by simply clicking the referred cells while typing a formula. The square bracket and the coordinate row and column numbers will be entered in the formula automatically.

Example:

An example of a formula that add the value of the cell in row 2 and column 2 to the value of the cell in row 3 and column 2.

```
= 10.5 + [2,2] + [3,2]
```

Formulas can also refer to cells in a different table.

Example:

A formula in Table2 can refer to a cell in Table1 as follows:

```
= 10.5 + [2,2] + Table1[3,3]
```

When a table refers to its own cell, it does not need to specify its own name; however, when it refers to a different table it does need to specify the other table's name.

Cell Selection

Contents of a Table cell can be selected by clicking on the start of the selection cell and dragging the mouse to the end of the selection cell.

Before selection can begin, it should be ensured that the cursor looks like ✚.

Selection can also be made using TUI. The syntax for single cell selection is

```
    selectCell(<row>, <col>)
```

`<row >` represents row number and `<col >` represents the column number of the cell to be selected.

The syntax for multiple cell selection is

```
    selectCell(<firstRow>, <firstCol>, <lastRow>, <lastCol>)
```

`<firstRow>` and **`<firstCol>`** represent the row and column numbers of the cell at the start of the block of the cells to be selected. **`<lastRow>`** and **`<lastCol>`** represent row and column numbers of the last cell of the block of cells to be selected.

Items in the angle brackets are supplied by the user.

Delete

Contents of the selected cells can be deleted by either selecting the [Delete] menu item from the Edit menu or by pressing the Delete key on the keyboard.

Delete operation can also be performed using TUI. The syntax for cut is

```
<Table Name>.delete()
```

Cut

Contents of a Table cell can be cut, copied, and pasted. These actions work similar to that of other spread sheet programs, such as Excel®.

Cut and Paste operations can also be performed using TUI. The syntax for cut is

```
<Table Name>.cut()
```

Copy

Like in other spread sheet programs, when copied and pasted the formula indices are modified to maintain relative references. For example, if a cell is copied then pasted in 3 cells down, the row indices in the formula (which are referring to other cells) will be increased by 3. Likewise, if the copied cells are pasted 2 cells to the right, the column indices will be increased by 2.

Pasted data will not be modified. To prevent certain indices from modifying, use a $ in front of the indices.

The TUI syntax for copy is

```
<Table Name>.copy()
```

Paste

And the syntax for paste is

```
<Table Name>.paste()
```

Move

Contents of the cells can be moved by either cutting and pasting or by dragging on the selected cells border. Before move can begin, it should be ensured that the cursor looks like ✥.

Fill Down Fill Right

Fill Down and Fill Right actions can be used to replicate a formula several times. To fill down, execute the following steps:

1. Select the desired cell
2. Move the cursor to the down-right corner of the selected cell. The cursor should change to +.
3. Click and drag down or right.

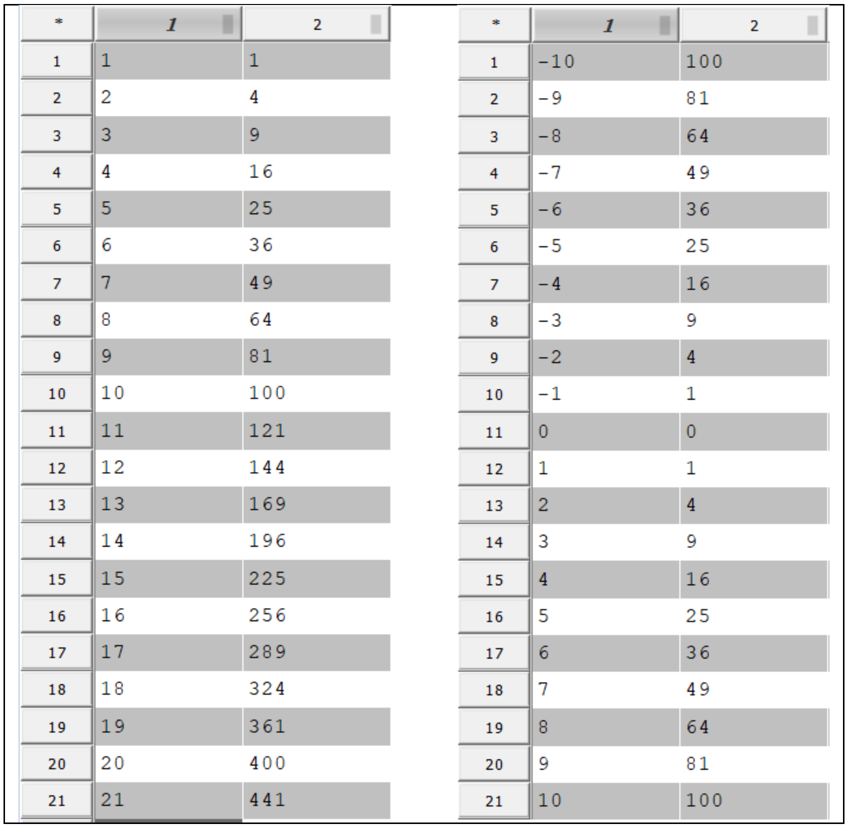
As with Copy and Paste operations, the indices in the formula of the replicated cell will be updated to maintain relative reference.

Example: Creating function datasets

The table app can be used to create table data that relates to a function. For this example, we will create a simple data set for x vs x^2.

1. Click on the cell[1,1] and enter the number 1.
2. Click on the cell[2, 1] and type in "=" to start a formula.
3. Click on the cell[1,1] or pressthe **Up Arrow, (↑)** key on the keyboard. This will make reference to the cell [1,1] in the formula as "= [1,1]".
4. Add " + 1" to the formula. Now the formula should look like "= [1,1] + 1". This mean the value in the cell[2,1] will be 1 increment of the value in the cell[1,1], which is 2.
5. Click on the bottom right corner of the cell[2,1] when the + icon appears and then drag down till you reach the number 21. This fills down the formula of the first cell onto the cells below.
6. Click on the cell[1, 2] and type in "=" to start another formula.
7. Click on the cell[1,1] or press the **Left key (←)** on the keyboard.
8. Add "^2" to the formula. This mean the value in the cell[1,2] will be square the value of the cell[1,1].
9. Click on the bottom right corner of the cell when the + icon appears and then drag down till you reach the number 441.
10. Column 1 has a data set of [1,21] with increments of 1, and column 2 has those values squared (Shown on the left side of the table below).
11. Change the value of the cell[1,1] to -10. This will update all the entrees. Now, Column 1 has a data set of [-10,10] with increments of 1, and column 2 has a data set of [100,100] (Shown on the right side of the table below).

Cell[1,1] = 1	**Cell[1,1] = -10**



*	1	2
1	1	1
2	2	4
3	3	9
4	4	16
5	5	25
6	6	36
7	7	49
8	8	64
9	9	81
10	10	100
11	11	121
12	12	144
13	13	169
14	14	196
15	15	225
16	16	256
17	17	289
18	18	324
19	19	361
20	20	400
21	21	441

*	1	2
1	-10	100
2	-9	81
3	-8	64
4	-7	49
5	-6	36
6	-5	25
7	-4	16
8	-3	9
9	-2	4
10	-1	1
11	0	0
12	1	1
13	2	4
14	3	9
15	4	16
16	5	25
17	6	36
18	7	49
19	8	64
20	9	81
21	10	100

Table App's Text User Interface (TUI)

The TUI for the Table app consists of functions and methods. Functions are independent of particular instances of Table app; whereas, the methods are applicable only to a instance of Table app. Therefore, methods are called with a dot notation, and the functions are called without it.

Syntex Function call:

```
<function name>(param1, param2, …, paramN)
```

The TUI functions for the Table app are described in the table below:

Function	Description
`createTable("<Table object Name">)`	Creates a new Table object in the selected document. The name of the object is provided by the user in the angle bracket (<>).

Syntax Method call:

```
<table name>.<method name>(param1, param2, …, paramN)
```

The TUI methods for the Table app are described in the table below (in alphabetical order):

Method	Description	Return
alignBottom()		
alignCenter()		
alignLeft()		
alignMiddle()		
alignRight()		
alignTop()		
appendColumn()		
appendColumn(integer cols)		
appendRow()		
appendRow(integer rows)		
bold()		
clear()		
clearAllSelection()		
copy()		
copy(integer row, integer col)		
copy(integer row1, integer col1, integer row2, integer col2)		
cut()		
cut(integer row, integer col)		
cut(int row1, int col1, int row2, int col2)		
decreaseDecimalPoint()		
decreaseDecimalPoint(integer num)		
delete()		
delete(integer row, integer col)		
delete(integer row1, integer col1, integer row2, integer col2)		
exportFile(String path)		
exportTable(String path)		
exportTableAs()		
fillColor()		
fillColor(integer red, integer green, integer blue)		
fillColor(integer red, integer green, integer blue, integer alpha)		
fillColor(real red, real green, Double blue)		
fillColor(real red, real green, real blue, real alpha)		
fillColor(String colorName)		

fillDown()		
fillLeft()		
fillRight()		
fillUp()		
fontFamily(String name)		
fontSize(integer size)		
get()		Table2d
getColumnCount()		integer
getColumnWidth()		
getColumnWidth(int col)		
getIndex(integer row, integer col)		object
getIndex(range row, range col)		object
getRowCount()		integer
getRowHeight()		integer
getRowHeight(integer row)		integer
getSelectedColumn()		integer
getSelectedColumnCount()		integer
getSelectedRow()		integer
getSelectedColumn()		integer
getSelectedColumnCount()		integer
getSelectedRow()		integer
getSelectedRowCount()		integer
importFile(String dirName, String fileName)		
importFile(String path)		
importTable(String dirName, String fileName)		
importTable(String path)		
increaseDecimalPoint()		
increaseDecimalPoint(integer num)		
insertColumns()		
insertColumns(integer col)		
insertColumns(integer colStart, integer cols)		
insertRows()		
insertRows(integer row)		
insertRows(int rowStart, integer rows)		
italic()		
lineColor()		
lineColor(integer red, integer green, Integer blue)		
lineColor(integer lineNum, integer red, integer green,		

integer blue, integer alpha)		
lineColor(real red, real green, real blue)		
lineColor(real red, real green, real blue, real alpha)		
lineColor(String colorName)		
createTable()		
paste()		
removeColumn()		
removeColumns(integer col)		
removeColumns(range col)		
removeColumns(realVector col)		
removeColumns(array col)		
removeRow()		
removeRows(integer row)		
removeRows(range row)		
removeRows(realVector row)		
removeRows(array row)		
select(real row, real col)		
select(integer firstRow, integer firstCol, integer lastRow, integer lastCol)		
selectAll()		
selectAllData()		
selectColumn(integer col)		
selectColumns(integer firstCol, integer lastCol)		
selectRow(integer row)		
selectRows(integer firstRow, integer lastRow)		
setAllBorders()		
setBorderNone()		
setBottomBorder()		
setBottomDoubleBorders()		
setColumnWidth(integer width)		
setColumnWidth(integer col, integer width)		
setIndex(integer row, integer col, HYP_Object value)		
setIndex(range row, range col, HYP_Object value)		
setLeftBorder()		
setOutsideBorder()		
setRightBorder()		
setRowHeight(integer height)		
setRowHeight(integer row, integer height)		
setSelectedRowHeight(int height)		

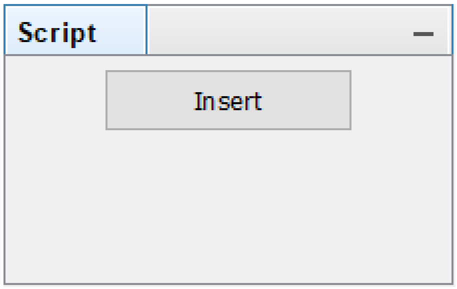
setThickBottomBorders()		
setThickBottomBorders()		
setThickBoxBorders()		
setTopAndBottomBorders()		
setTopAndBottomBorders()		
setTopAndDoubleBottomBorders()		
setTopAndThickBottomBorders()		
setTopBorder()		
textColor()		
textColor(integer red, integer green, integer blue)		
textColor(integer red, integer green, integer blue, integer alpha)		
textColor(real red, real green, Double blue)		
textColor(real red, real green, real blue, real alpha)		
textColor(String colorName)		
textStyle(boolean bold, boolean italic)		
trim()		

6.3.5 Using Script App

Scripts are used to perform preprogrammed tasks. Scripts can automate tasks that are complex and repetitive by combining different commands within the structure of programming language. LDV Scripts are written using an interpretive language, Hyper. Refer to the Hyper Reference Manual for detail. Example scripts can be found in the **script folder**.

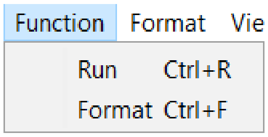
Tool Panel

The tool panel for this app is shown below:

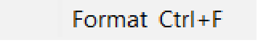


Menu Bar

The Script app uses the standard app menu bar. However, the Function menu has items specific to the Table app. The Function menu for this app is described below:



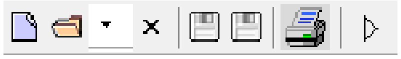
The menu items for the Function menu are described in the table below:

Menu Item	Description
	Executes the script. The keyboard shortcut is **CTRL+R**.
	Formats the script by properly indenting according to syntax structure and color codding keywords, data, comments, etc. The keyboard shortcut is **CTRL+F**.

Toolbar

The Script app uses the standard app toolbar with one extra tool, the **[Run]** button.

The Script app tool bar is shown below:



The **[Run]** button of the Script app toolbar is described in the table below:

Icon	Name	Description
▷	Run	Executes the script.

Inserting new Script component in a document

A new Script component can be inserted in a document using GUI or command. The following steps will insert a Script component using GUI:

1. Select the **[Insert]** button from the Script tool pane.
2. Either click or click and drag in the selected open document.

If just clicked, a Script app frame of a predetermined size will be created and embedded in the document. If clicked and dragged, the app frame size will be determined by the drag action.

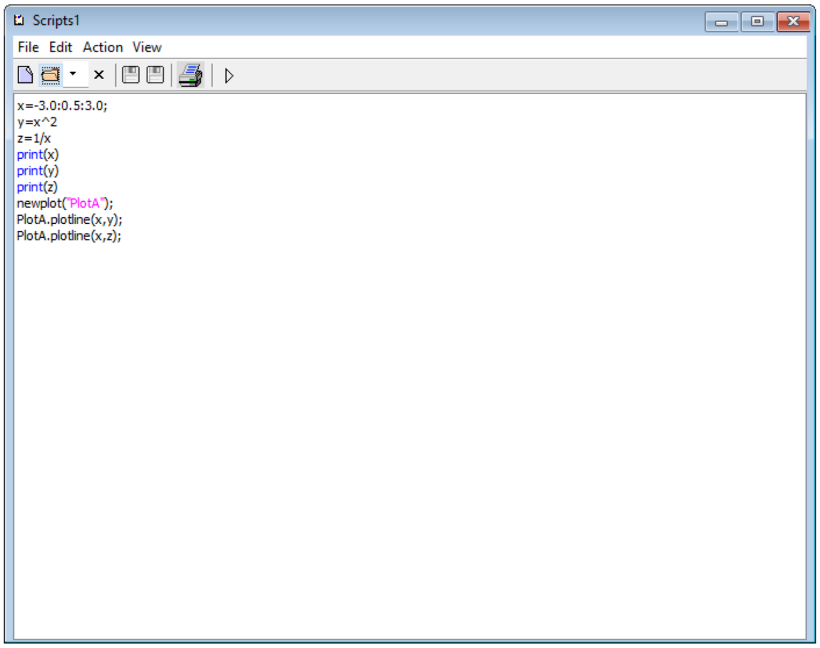
A Script component can also be inserted using the following command:

```
createScript(<script object name>)
```

The string inside the angle bracket (< >) is the name the user wants to assign to the inserted component.

Once a Script component is inserted, new script can be added, edited, and formatted or script can be imported from a file.

An example of a Script component depicted below:



```
x=-3.0:0.5:3.0;
y=x^2
z=1/x
print(x)
print(y)
print(z)
newplot("PlotA");
PlotA.plotline(x,y);
PlotA.plotline(x,z);
```

Editing

Editing scripts is similar to using any text editor.

The TUI methods for the Table app are described in the table below (in alphabetical order):

Method	Description	Return
`alignBottom()`		
`alignCenter()`		

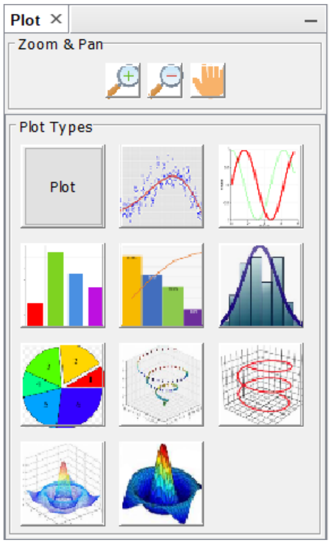
6.3.6 Using Plot App

Plots are used to graphically represent numerical data. The Plot app can be used to produce several different types of plots. The types of plots the Plot app can produce are listed in the table below:

2-D Plot	3-D Plot
Scatter Plot	Scatter Plot
Line Plot	Line Plot
Bar Chart	Mesh Plot
Pie Chart	Surface Plot
Histogram	
Pareto Chart	
Bode Plot	
Root Locus Plot	
Step Response Plot	

Tool Panel

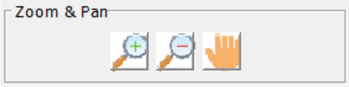
The tool panel for this app is shown below:



The Plot tool panel is organized in two groups:

1. Zoom & Pan
2. Plot Types

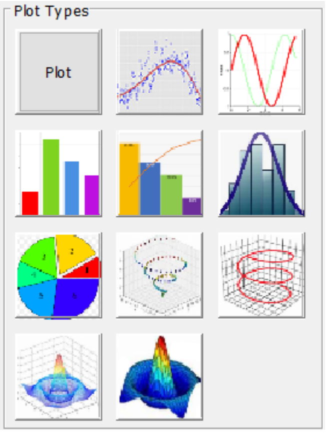
The Zoom & Pan group contains tools related to zooming and panning and is depicted below:



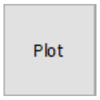
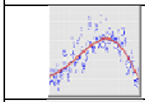
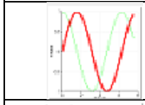
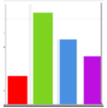
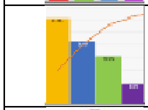
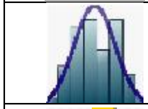
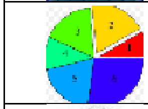
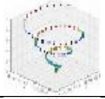
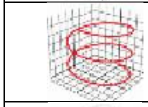
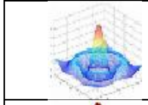
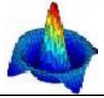
The items in the Zoom & Pan group are described in the table below:

Icon	Name	Description
🔍	Zoom In	Used for zooming in the plot. After selecting the tool, a single click on the plot will zoom in by 10%, pressing the left mouse button and dragging will draw a zoom box and releasing the left mouse button will zoom in the plot to the zoom box. Double clicking will restore to the original zoom level.
🔍	Zoom Out	Used for zooming out. After selecting the tool, a single click on the plot will zoom out be 10%
✋	Pan	Used for panning (shifting) the plot. After selecting the tool, pressing the left mouse button, and dragging and releasing the left mouse button will shift the plot by the amount the mouse was dragged.

The Plot Types group contains tools creating different type of plots and is depicted below:



The items in the Plot Types group are described in the table below:

Icon	Name	Description
	Generic Plot	Used to create an empty Plot app frame on a document. After selecting the tool, clicking or dragging on a document will insert a Plot app frame on the document. Clicking will produce a predetermined sized frame and dragging will set the set the frame size according to the amount dragged.
	2D Scatter Plot	Used to insert a 2D scatter plot using the selected data from a table.
	2D Line Plot	Used to insert a 2D line plot using the selected data from a table.
	Bar Chart	Used to insert a bar chart using the selected data from a table.
	Pareto Chart	Used to insert a pareto chart plot using the selected data from a table.
	Histogram	Used to insert a histogram using the selected data from a table.
	Pie Chart	Used to insert a pie chart plot using the selected data from a table.
	3D Scatter Plot	Used to insert a 3D scatter plot using the selected data from a table.
	3D Line Plot	Used to insert a 3D line plot using the selected data from a table.
	3D Mesh Plot	Used to insert a 3D mesh plot using the selected data from a table.
	3D Surface Plot	Used to insert a 3D surface plot using the selected data from a table.

A script is needed to create a plot within the document. There are several available plot types, including line, scatter, histogram and mesh for example. The examples below will help guide users to create custom plots, as necessary.

Plot Adornment

Visual options for the plot include changing the background color, the grid color, and linewidth of plot.

Function	**Input type**	**Function description**
Axis Labels		
xLabel(label)	String label	Adds an x-axis label to plot
xLabelFont(name)	String name	Changes the font of the x-axis label
xLabelSize(size)	Integer size	Changes the size of the x-axis label
xLabelStyle(style)	Integer style	Changes the style of x-axis label
xLabelStyle(colorName)	String colorName HYP_JavaValue color	Changes the color of the x-axis label
xAxisColor(colorName)	String colorName HYP_JavaValue color	Changes the color of the x-axis
yLabel(label)	String label	Adds a y-axis label to plot
yLabelFont(name)	String name	Changes the font of the y-axis label
yLabelSize(size)	Integer size	Changes the size of the y-axis label
ylabelStyle(style)	Integer style	Changes the style of y-axis label
ylabelStyle(colorName)	String colorName HYP_JavaValue color	Changes the color of the y-axis label
yAxisColor(colorName)	String colorName HYP_JavaValue color	Changes the color of the y-axis
zlable(label)	String label	Adds an z-axis label to plot
zlabelFont(name)	String name	Changes the font of the z-axis label
zlabelSize(size)	Integer size	Changes the size of the z-axis label
zlabelStyle(style)	Integer style	Changes the style of z-axis label
zlabelStyle(colorName)	String colorName HYP_JavaValue color	Changes the color of the z-axis label
zAxisColor(colorName)	String colorName HYP_JavaValue color	Changes the color of the z-axis
axis(xmin, xmax, ymin, ymax)	double, double, double, double	Sets maximum and minimum values for x and y-axis
Plot Legend		
showLegend()		Adds legend to plot
hideLegend()		Hides legend
legend(aFlag)	Boolean aFlag	Adds legend with a boolean
dataName (dataNum, str)	Integer data Num, String str String data Num, String str	Sets name for specific data set
dataName(list)	HYP_ArrayList list	{“a”, “b”,”c”},

dataName(listID, listVal)	Hyp_ArrayList list ID, HYP_ArrayList listVal	
legendFont(name)	String name	Changes font of legend
legendSize(size)	Integer size	Changes size of legend
legendStyle(style)	Integer style	Changes style of legend
legendColor(colorname)	String colorName HYP_JaveValue color	Changes color of legend
legendBorderColor(colorname)	String colorName HYP_JaveValue color	Changes border color of legend
legendBackgroundColor(color)	String colorName HYP_JaveValue color	Changes background color of legend
Plot Title		
title(str)	String str	Adds title to plot
getTitle()		
titleFont(name)	String name	Changes font of title
titleSize(size)	Integer size	Changes size of title
titleStyle(name)	Integer size	Changes style of title
titleColor(colorName)	String colorName HYP_JaveValue color	Changes color of title
titleBackgroundColor(color)	String colorName	Change color of title background
Plot Annotations		
annotate(label)	String label	Adding annotation to plot Goes to default location, can be dragged
annotate(label, x, y)	String label, Integer x, Integer y String label, Long x, Long y	Define location of annotation box (box can be dragged to desired location
annotateFont(name)	String name	Changes annotation font
annotationSize(size)	Integer size	Changes annotation size
annotationStyle(style)	Integer style	Changes annotation style
annotationColor(colorname)	String colorName HYP_JaveValue color	Changes the color of the word in the annotation
annotationBackgroundColor(color)	String colorName HYP_JaveValue color	Changes annotation background color
Plot Grids		
majorGridColor(colorname)	String colorName HYP_JaveValue color	Changes the color of the major gridline
minorGridColor(colorname)	String colorName HYP_JaveValue color	Changes color of minor gridline
gridColor(colorName)	String colorName HYP_JaveValue color	Changes color of grid on plot
setGridColor(------------------------)		

`gridOn()`		Adding gridlines on plot
`gridOff()`		Hiding gridlines on plot
`horGridOn()`		Adding horizontal gridlines
`horGridOff()`		hiding horizontal gridlines
`verGridOn()`		Adding vertical gridlines
`verGridOff()`		Hiding vertical gridline
Point/Line Adornment		
`lineColor(linenum, red, green, blue)`	Integer lineNum, integer red, green, blue Integer lineNum, double red, green, blue Integer lineNum, String colorName Integer lineNum, HYP_JavaValue	Changes line color using line number Integer input 0 - 255
`lineColor(lineName, red, green, blue)`	String lineName, integer red, green, blue String lineName, double red, green, blue String lineName, String colorName String lineName, HYP_JavaValue	Changes line color using line name Integer input 0-255
`lineColor(list)`	HYP_ArrayList list	
`lineColor(listID, listVal)`	HYP_ArrayList listID, HYP_ArrayList listVal	
`pointColor(pointNum, red, green, blue)`	Integer pointNum, integer red, green, blue Integer pointNum, double red, green, blue	Changes color of point using point number and color fraction
`pointColor(pointNum, red, green, blue, alpha)`	Integer pointNum, integer red, green, blue, alpha Integer pointNum, double red, green, blue, alpha	Changes point color using point number Integer input 0 - 255
`pointColor(pointNum, colorName)`	Integer pointNum, String colorName Integer pointNum, HYP_JaveValue color	Changes point color using point number and color name
`pointColor(pointName, red, green, blue)`	String pointName, integer red, green, blue String pointName, double red, green, blue	Changes point color using point name Integer input 0-255
`pointColor(pointName, red, green, blue, alpha)`	String pointName, integer red, green, blue, alpha String pointName, double red, green, blue, alpha	Changes point color using point name Alpha = makes something transparent (example alpha = 50, 50% transparent)
`pointColor(pointNum, colorName)`	String pointName, String colorName HYP_JavaValue color	
`pointColor(listID, listVal)`	HYP_ArrayList listID, HYP_ArrayList listVal	
`lineType(lineNum, type)`	Integer lineNum, String type String lineNum, String type	
`lineType(type)`	String type	Changes line type
`lineType(list)`	HYP_ArrayList list	
`lineType(listID, listVal)`	HYP_ArrayList listID, HYP_ArrayList listVal	

pointType(pointNum, type)	Integer pointNum, String type	Changes point type using point number
pointType(pointName, type)	String pointName, String type	Changes point type using point name
pointType(list)	HYP_ArrayList list	?
pointType(listID, listVal)	HYP_ArrayList listID, HYP_ArrayList listVal	?
lineWidth(lineNum, width)	Integer lineNum, Integer width Integer lineNum, Double width	Changes line width sing line number
lineWidth(listID, listVal)	HYP_ArrayList listID, HYP_ArrayList listVal	
pointSize(pointNum, size)	Integer pointNum, Integer size	Changing point size using point number
pointSize(pointNum, width)	Integer pointNum, Double width	Changing point width using point number
pointSize(pointName, width)	String pointName, integer width String pointName, double width	Changing point width using point name
explode()		
explode(explodes)	Integer explodes Double explodes HYP_RealVector explodes	
setPieCircle()		
setPieOval()		
Plot Adornment		
setBackgroundColor(colorName)	String colorName	
Plot type		
SetPlotType(PlotType plotType		Which function name??

Color

There are 3 ways to describe the desired color of an element: name, integer, percent. Names of the colors that can be used are listed below in table ____.

To describe a color using integers, amounts of blue, red, and green from 0 to 255 are chosen. For example, purple is 112 blue, 113 red, and 0 green.

Describing a color using fractions (or percentages) is similar to using integers. Amounts of blue, red, and green from 0.0 to 1.0 are inputted. Purple is made up of 50% blue, 50% red, and 0% green.

Items on a plot with the ability to change colors include plot background, grid color, axis lines, and all fonts.

Function	**Input type**	**Function description**
General commands		

Function	Plot Types
PlotType	Scatter Line Bode Root Locus Step Bar Pareto Histogram Pie Scatter Line Mesh Surface
PlotType2D	Scatter Line Bode Root Locus Step Bar Pareto Histogram+ Pie
PlotType3D	Scatter Line
PlotTypeSurf	Mesh Surface

Sample scripts that uses the functions include:

plotline2.hyp

lp1.hyp

dp.hyp

6.3.6.1 Line Plot Example

An example of a line plot is given below.

```
x=-3.0:0.5:3.0;

y=x^2

z=1/x

print(x)

print(y)

print(z)

createPlot("PlotA");

PlotA.plotLine(x,y);

PlotA.plotLine(x,z);
```

This code can also be found in the scripts folder in the file named “plotline2.hyp”.

In this piece of code, the variables are first defined for the software to read. The independent variable here is x, which is defined using the format:

```
x=<left endpoint>:<increment level>:<right endpoint>;
```

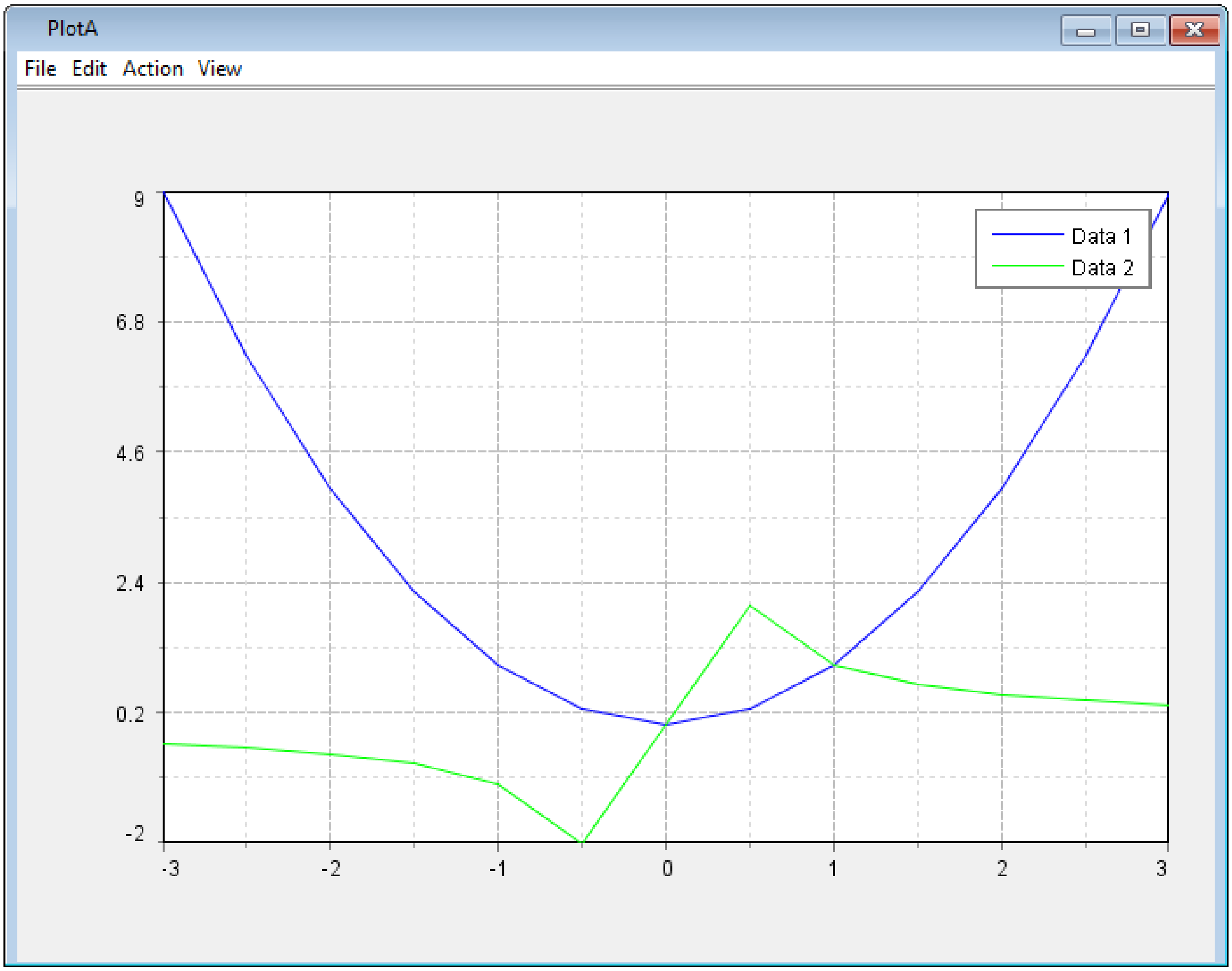
The **x** values here start at -3.0 and end at 3.5, incrementing by 0.5.

There are two dependent variables, y and z, which are defined after the independent variable x is defined.

The print function is not necessary to plot the data but used to display the entered data in an array format in the console.

The command **`createPlot("PlotA")`** is used to create a new plot window called “PlotA”. This is the plot window with axes where the data will be plotted.

The command **PlotA.plotLine(x,y);** is used to plot the variables x and y in the plot window PlotA, with x in the horizontal axis and y in the vertical axis. A similar line is used to plot x and z. The result will be seen in the screenshot below.



Data 1 is the first line, which in this case is x vs y.
Data 2 is the second line, which in this case is x vs z.

To make the graph smoother, the increment level would have to be decreased to make the data points closer together. Since this is a line plot, a discontinuous function like 1/x would still have the points connected as shown above.

6.3.6.2 Line Plot Decoration Example

After plots are drawn, they can be decorated using colors and symbols. The decorations work independently from the data. Both lines and plot windows can be decorated, and they also work independently of each other.

6.3.6.2.1 Line Decoration

An example of a line decoration script is given below.

```
exec("dataline")
createPlot("PlotA")
PlotA.plotLine(x,y)
PlotA.plotLine(x,z)
PlotA.lineWidth(2,5)
PlotA.title("Title")
PlotA.xLabel("Year")
PlotA.yLabel("Doller")
PlotA.lineWidth(1,5)
PlotA.lineType(1,"dash_dot")
PlotA.lineType(2,"solid")
PlotA.lineColor(1,"magenta")
PlotA.pointType(1,"star")
PlotA.pointType(2,"diamond")
PlotA.pointSize(1,25)
PlotA.pointSize(2,25)
PlotA.pointColor(1,"blue")
PlotA.pointColor(2,"red")
```

This code can also be found in the scripts folder in the file named “lp1.hyp”.

In this plot, PlotA is first plotted using data previously entered – in our case, it is plotline2.hyp.

The general formatting for line decoration is

```
<PlotLetter>.<function>(<parameters>)
```

The functions seen in this graph are as follows:

```
title (“<title>”)
```

Creates a title for the entire plot.

```
xLabel (“<label>”)
```

Labels the horizontal axis for the plot.

```
yLabel (“<label>”)
```

Labels the vertical axis for the plot.

```
lineType (<plot number>, <type>)
```

Changes the type of a specified plot line. Examples: solid, dash, dash_dot.

```
lineWidth (<plot number>, <width>)
```

Changes the width of a specified plot line. A higher number means a greater width.

```
lineColor (<plot number>, <color>)
```

Changes the color of a specified plot line.

```
pointType (<plot number>, <type>)
```

Changes the point type of a specified plot line. Examples: circle, star, diamond.

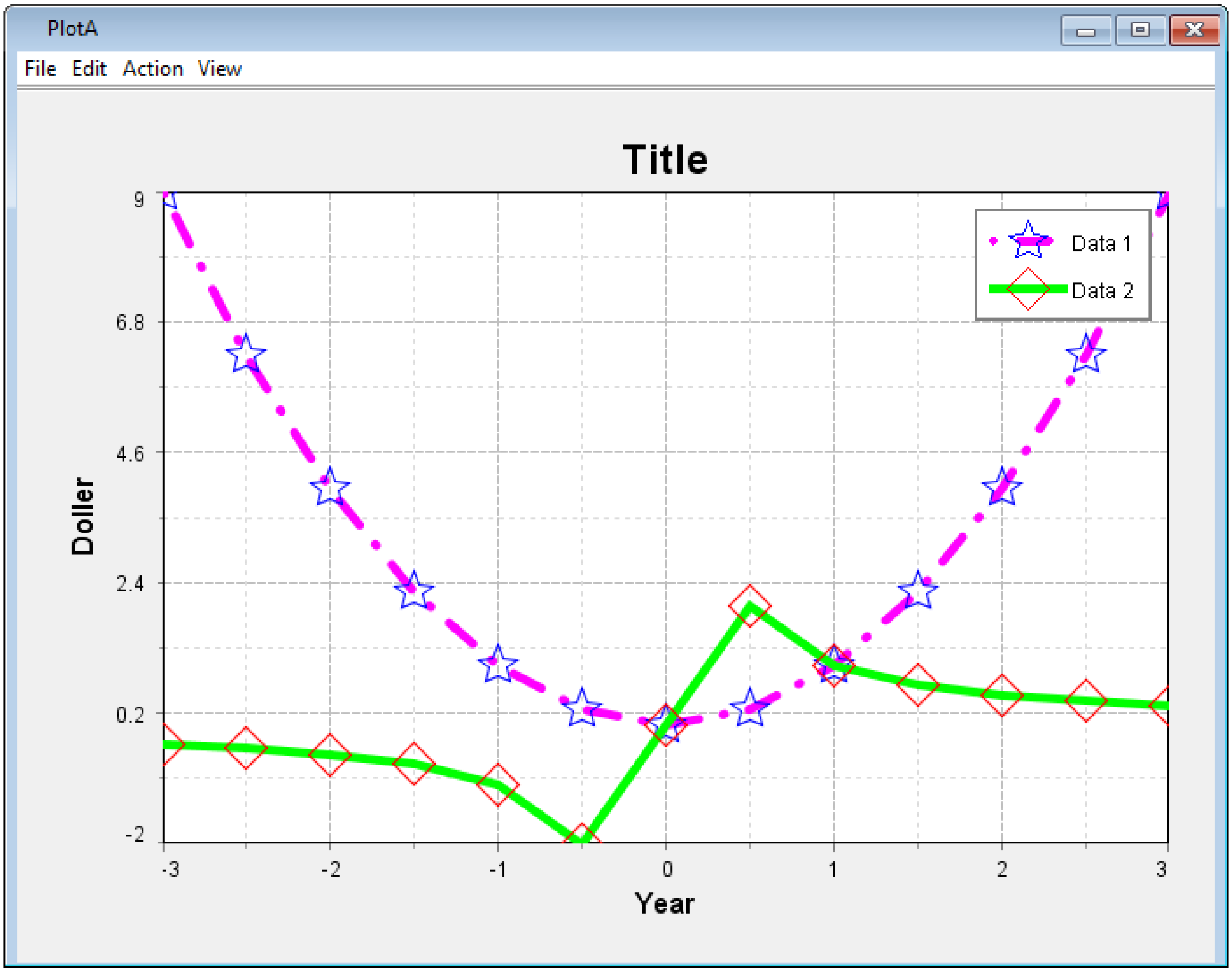
```
pointSize (<plot number>, <size>)
```

Changes the point size of a specified plot line. A higher number means a greater size.

```
pointColor (<plot number>, <color>)
```

Changes the point color of a specified plot line.

The functions and parameters in the code output a plot that looks like the screenshot below.



6.3.6.2.2 Plot Decoration

An example of a plot decoration script is given below.

```
PlotA.titleSize(36)

PlotA.titleFont("Algerian")

PlotA.titleColor("blue")


PlotA.xLabelSize(30)

PlotA.xLabelFont("forte")

PlotA.xLabelStyle("red")


PlotA.yLabelSize(24)

PlotA.yLabelFont("broadway")

PlotA.yLabelStyle("green")


PlotA.backgroundColor("pink")

PlotA.gridColor("cyan")

PlotA.minorGridColor("yellow")
```

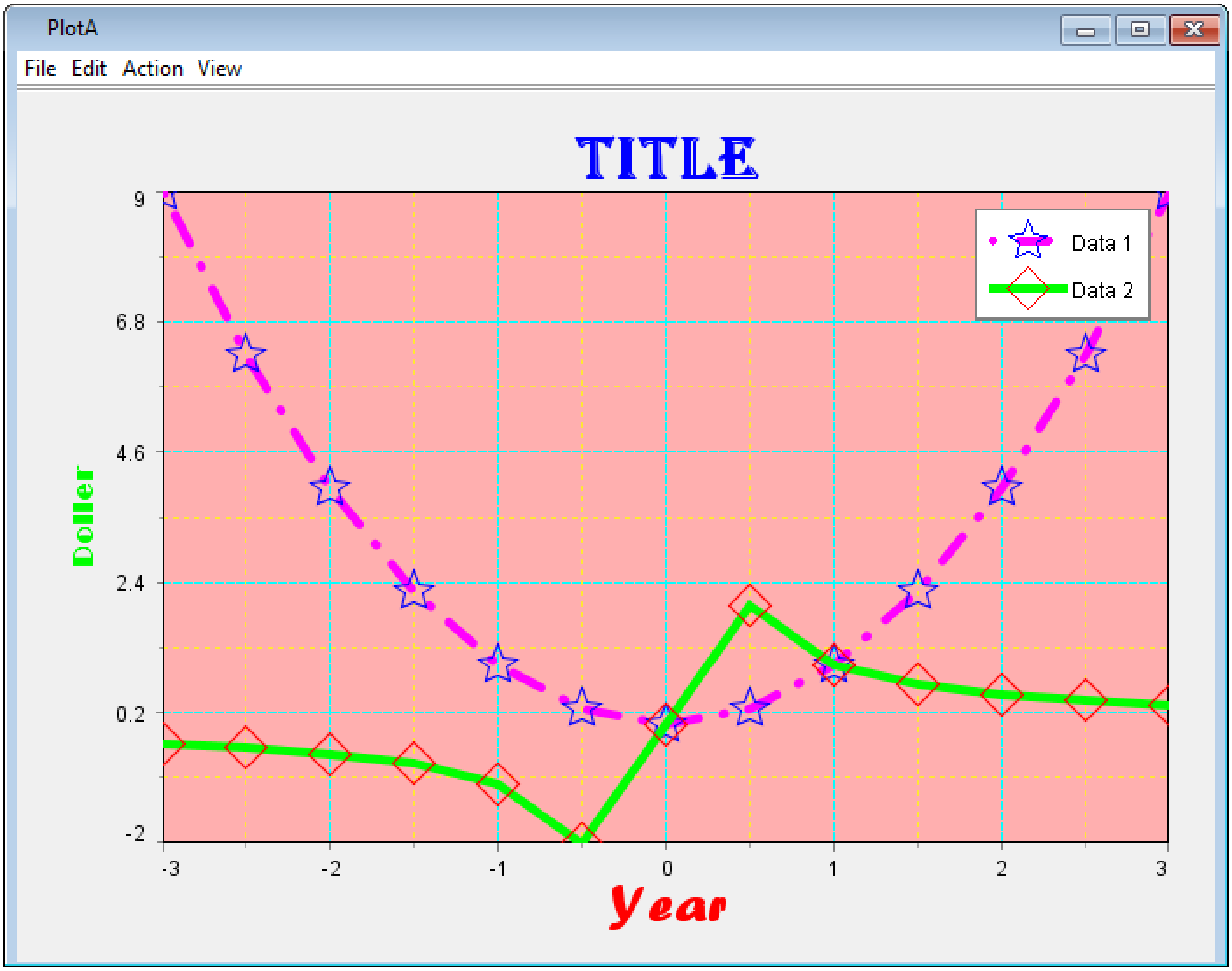
This code can also be found in the scripts folder in the file named “dp.hyp”.

As seen in the code, adding “size”, “font” or “color” after the title, xLabel and yLabel functions seen previously changes the size, font and color of those elements of the plot, by specifying the parameters in parentheses.

“PlotA.backgroundcolor” changes the background color of Plot A to pink, in the parentheses.

“PlotA.gridcolor” changes the grid color of Plot A to cyan, and “PlotA.minorGridcolor” changes the minor grid color of Plot A to yellow.

The results of the code are seen in the screenshot below.



The line decorations are carried over from the previous program, and have no bearing on the plot decoration script.

6.3.6.3 Multiple Plots & Scatter Plot Examples

6.3.6.3.1 Scatter Plot

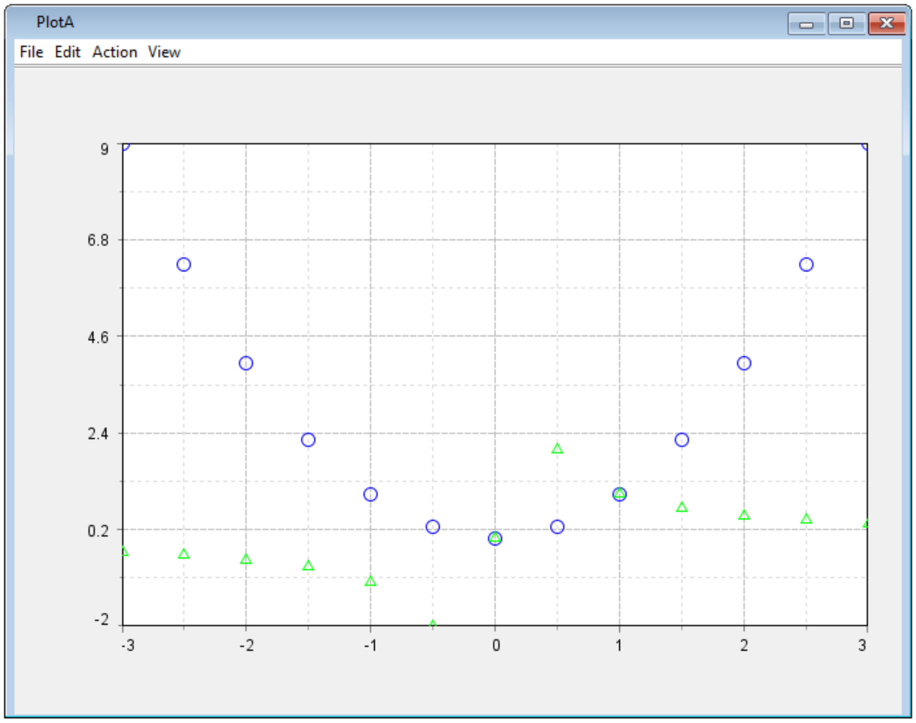
A scatter plot is very similar to a line plot. An example of a scatter plot is given below:

```
x=-3.0:0.5:3.0;

y=x^2

z=1/x

createPlot("PlotA");

PlotA.plotScatter(x,y);

PlotA.plotScatter(x,z);
```

It is seen that the code is very similar to that of a line plot, with the only difference being that the code ".plotLine" is replaced with ".plotScatter".

The software still uses variable data to plot the points in the same way as a line plot. The difference is visual – instead of joining the points, the points are plotted without having any lines to join them.

The code above creates an output that looks like the picture in the next page.



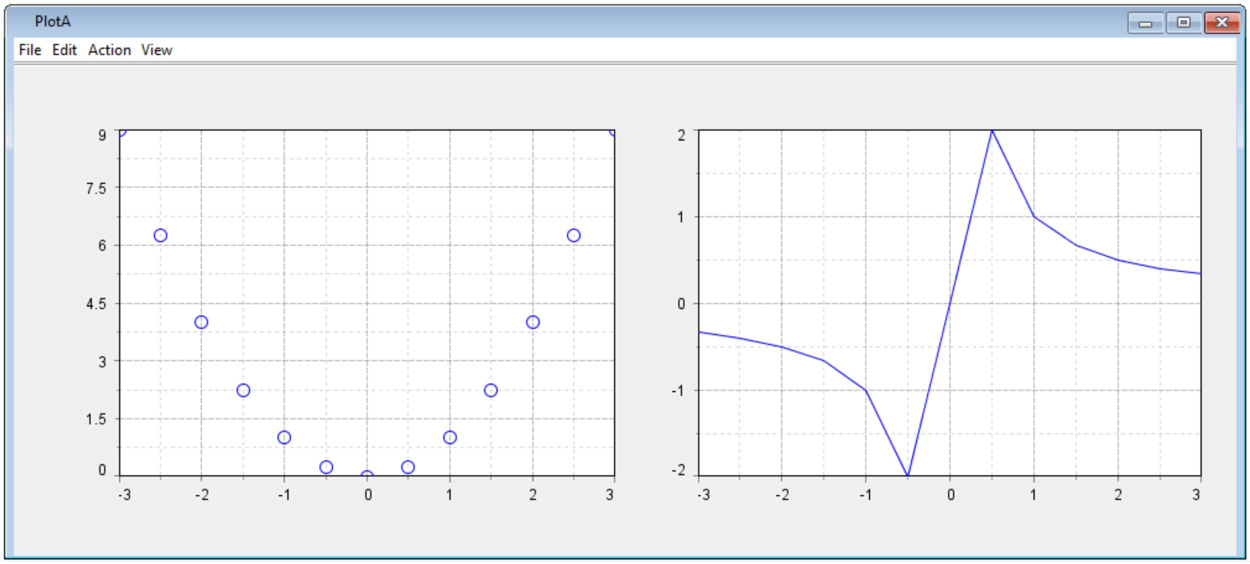
6.3.6.3.2 Multiple Plots

Multiple plot diagrams can be drawn in the same window. An example is seen below.

```
x=-3.0:0.5:3.0;

y=x^2

z=1/x

createPlot("PlotA");

PlotA[1,1].plotScatter(x,y);

PlotA[1,2].plotLine(x,z);
```

The difference seen here is the added `[ <integer> , <integer> ]` added immediately after “PlotA”. The software treats the plot window PlotA as a grid with rows and columns. The first integer corresponds to the row number and the second integer corresponds to the column number. In the above example, `[1,2]` corresponds to a plot being drawn in the first row and the second column.

The code above outputs a plot window that looks like this:



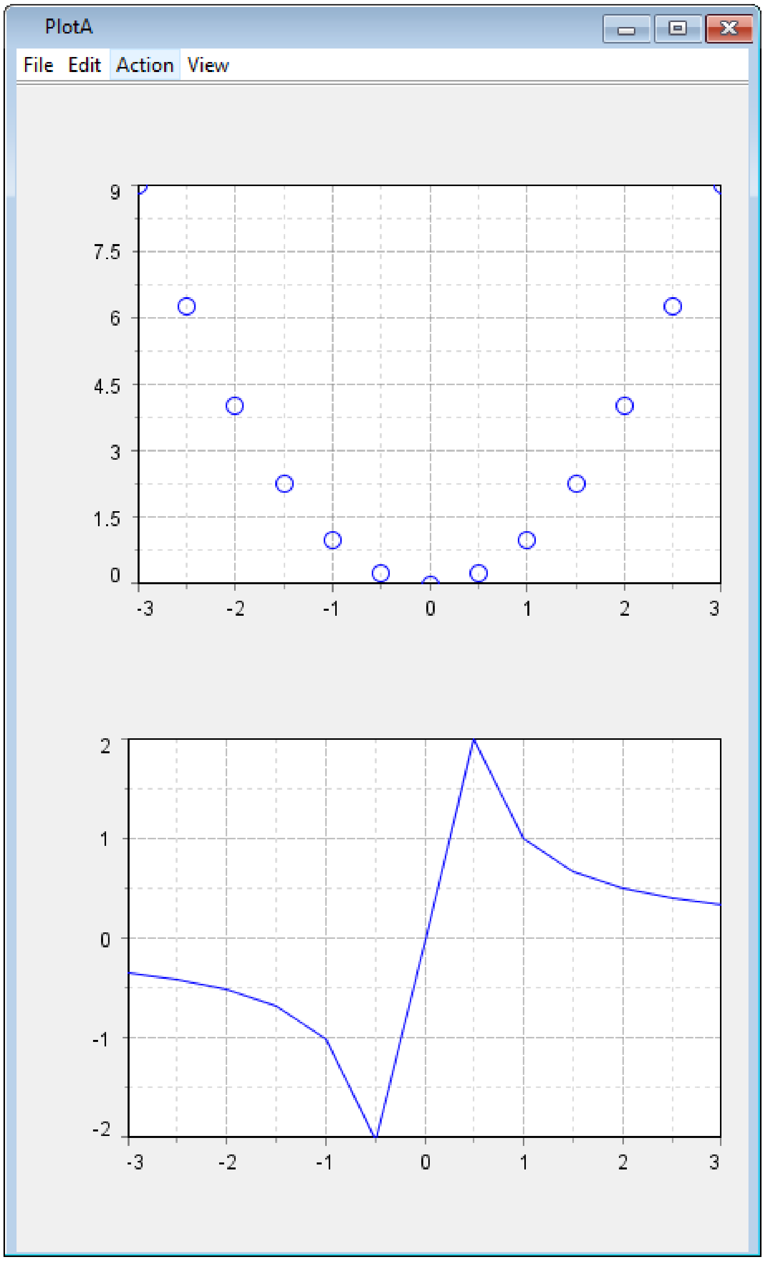
If the plots were to be stacked on top of one another instead of plotted side by side, the location for the second plot would be changed from [1,2] to [2,1]

```
PlotA[1,2].plotLine(x,z);
```

would be changed to:

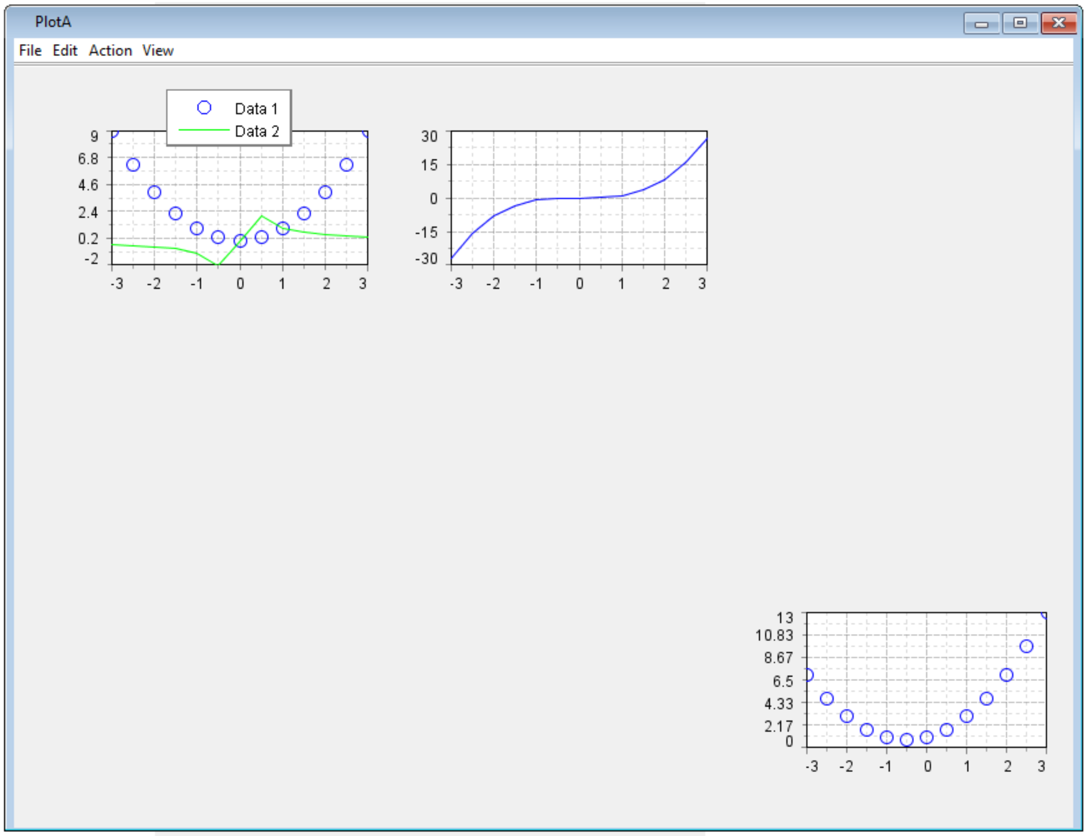
```
PlotA[2,1].plotLine(x,z);
```

This means that the plotLine would be drawn in the second row and the first column of the "grid", as seen below.



While the above examples show very “structured” examples where there are no empty locations on the grid, it does not necessarily have to be so. There can also be multiple plots in the same graph as seen in previous examples, while also being different plot types. An example of a “complex” plot window is shown below with its given code.

```
x=-3.0:0.5:3.0;

y1=x^2

y2 =x^3

z1=1/x

z2=x^2 + x + 1

createPlot("PlotA");

PlotA[1,1].plotScatter(x,y1);

PlotA[1,1].plotLine(x,z1);

PlotA[1,2].plotLine(x,y2);

PlotA[3,3].plotScatter(x,z2);
```



6.3.6.4 Bar Plot Example

A bar plot is used to plot numerical data against string labels. An example of a bar plot is given below.

```
bar_y = [10,20,30,40,50,60,75,55,45,25,15,5]

bar_x =
{"abc","def","ef","gh","ijkl","mno","pqr","st","uvw",
"xyz", "a123", "b456"}

createPlot("bar")

bar.plotBar( bar_x, bar_y )

bar.title("Bar Plot")
```

This code can also be found in the scripts folder in the file named “plotbar.hyp”.

“**bar_y**” is a variable created with a list that contains the data points for the vertical axis. The list of data points for the vertical axis must always be integer values. The elements in the list must be bounded by [ ].

“**bar_x**” is a variable created with a list that contains the data points for the horizontal axis. The list of data points for the horizontal axis must always be integer values. The elements in the list must be put in “ “ (quotation marks) and the elements must be bounded by { }.

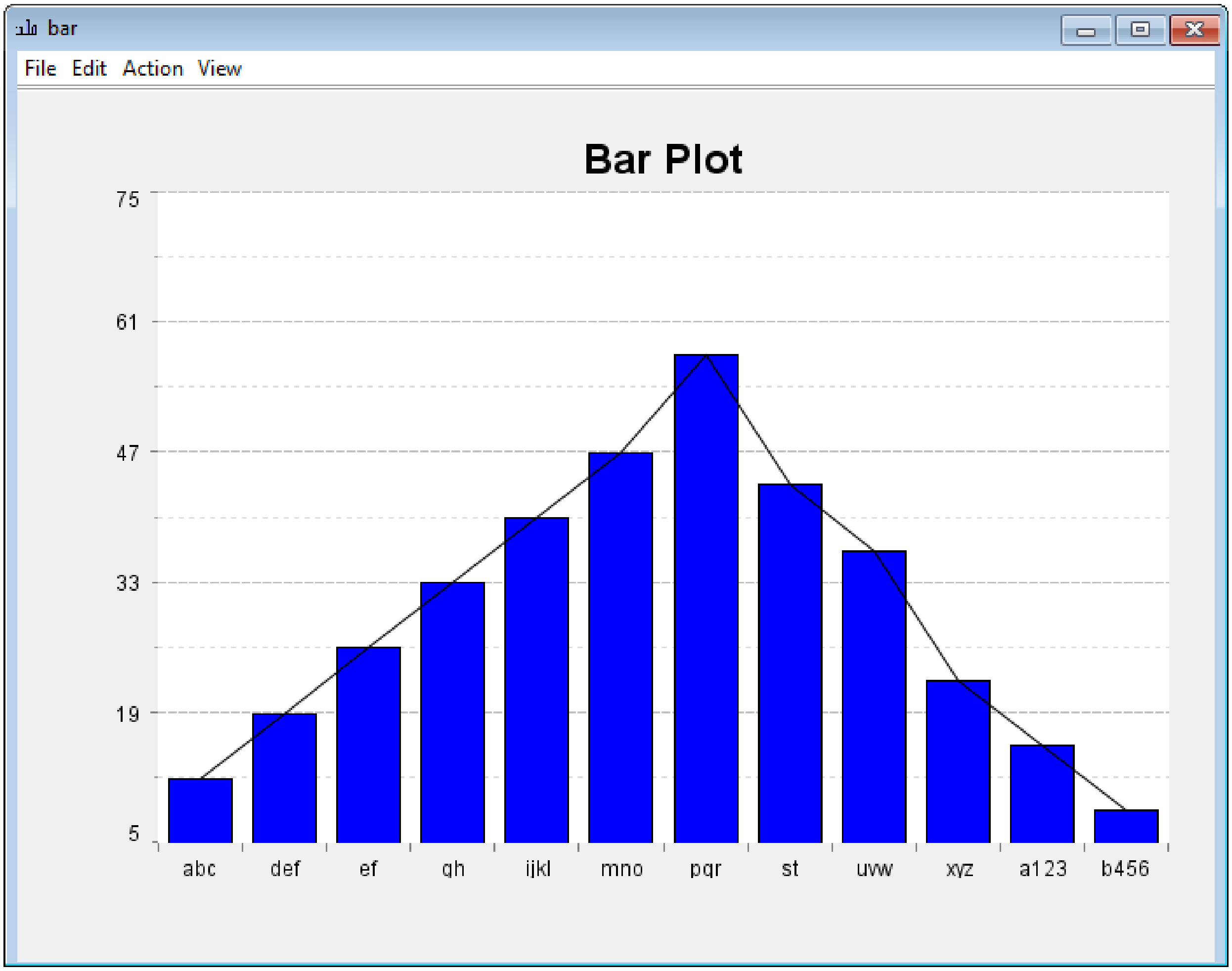
The vertical and horizontal variables must be equal in length (i.e. having the same number of elements) for the bar to be drawn. The data will be plotted exactly as typed in the list. For example, the list **{“ab”, “cd”, “ef”}** will be plotted with ab as the leftmost bar label, cd as the middle, and ef as the rightmost bar. The list of numbers will correspond directly to the list of strings. The list **[2, 4, 6]** will have **2** represent ab, **4** represent cd, and **6** represent ef.

The command “**createPlot(“bar”)**” creates a new plot window titled “bar”.

The command “**bar.plotBar(bar_x, bar_y)**” plots **bar_y** against **bar_x** in a bar plot. In the parentheses, the first variable is always plotted on the horizontal axis and the second variable is plotted on the vertical axis. The overall format is always “**bar.plotBar(<horizontal variable>,<vertical variable>)”.**

The command “**bar.title(“Bar Plot”)**” titles the plotted data as “Bar Plot”.

The result of this piece of code can be seen in the following page.



This is the result of the code seen in the previous page.

6.3.6.5 Pareto Plot Example

A pie plot is used to plot numerical data in a bar chart with descending numbers, and a cumulative percentage line plot on top.

```
y=[10,20,30,40,50,40,30,20,10]

x=[1,2,3,4,5,6,7,8,9]

l={"a","b","c","d","e","f","g","h","i"}

createPlot("pareto")

pareto.plotPareto(l,y)

pareto.title("Pareto Plot")
```

This code can be found in the scripts folder in the file named "plotpareto.hyp".

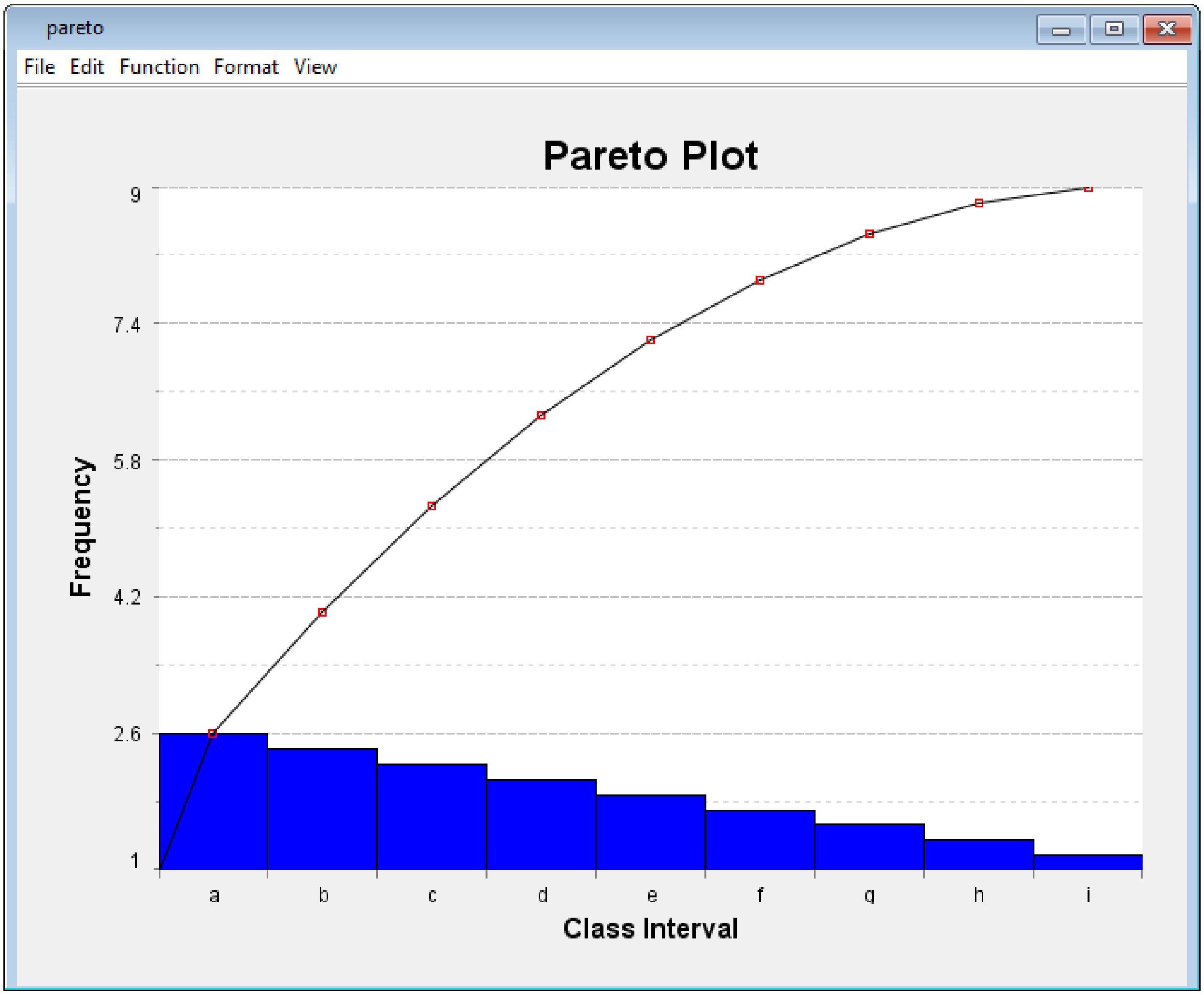
The command "**`y=[10,20,30,40,50,40,30,20,10]`**" and "**`x=[1,2,3,4,5,6,7,8,9]`**" create lists of numerical data points that will be plotted. The frequencies and cumulative percentages are calculated automatically and are seen on the chart.

The command "**`l={"a","b","c","d","e","f","g","h","i"}`**" creates a list of labels for the bars on the plot. They are fixed – any numerical changes in the data set will not change the order of the labels.

The command "**`createPlot("pareto")`**" creates a new plot window named "pareto".

The command "**`pareto.plotPareto(l,y)`**" creates a pareto chart in the "pareto" plot window using the specified parameters. "y" is the list of data points that will be plotted on the chart. "l" is the list of labels used for the bar chart. The command "**`pareto.title("Pareto Plot")`**" labels the plot with the title "Pareto Plot".

The end result looks like the image below.



6.3.6.6 Histogram Plot Example

A histogram plot is used to plot the distribution of numerical data. An example of a histogram plot code is given below.

```
y=[10,20,30,40,50,40,30,20,10]

x=[1,2,3,4,5,6,7,8,9]

createPlot("hist")

hist.plotHist(x,y)

hist.title("Histogram")
```

6.3.6.7 Pie Plot Example

A pie plot is used to plot numerical data on a pie chart as percentages. An example of a pie plot code is given below.

```
x=[100,200,400,500]

createPlot("pie")

pie.plotPie(x, true, 10.0)
```

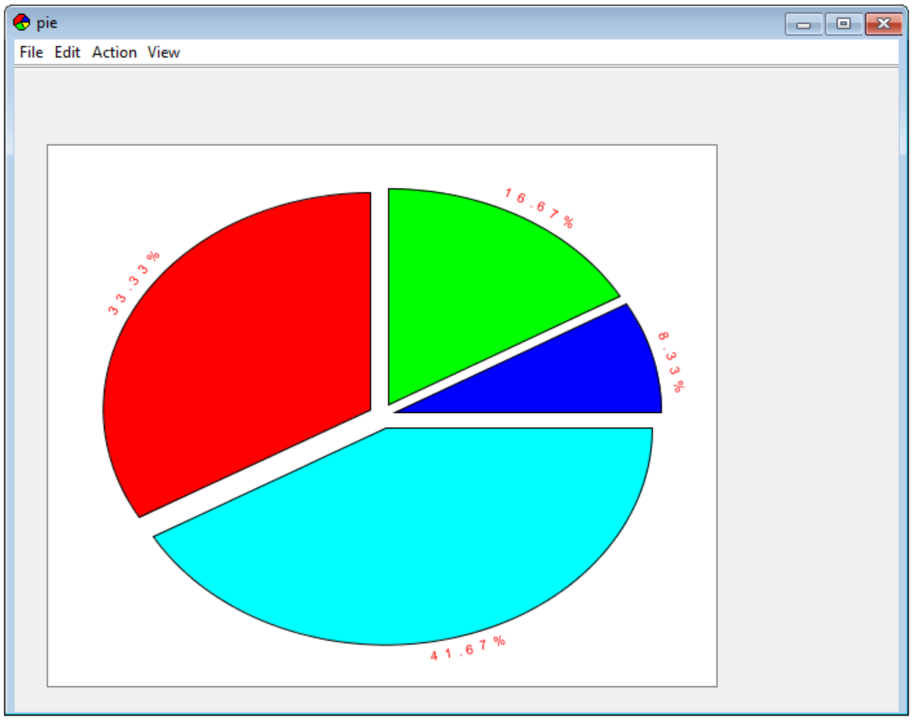
This code can be found in the scripts folder in the file named “plotpie.hyp”.

The command “**`x=[100,200,400,500]`** creates a list of numerical data points that will be plotted on the chart. The percentages that are seen on the chart are calculated automatically. There can be a maximum of 12 data points, after which an incomplete pie chart is plotted using the first 12 specified data points.

The command “**`createPlot(“pie”)`**” creates a new plot window named “pie”.

The command “**`pie.plotPie(x, true, 10.0)`**” creates a pie chart in the “pie” plot window using the specified parameters. “**`x`**” is the list of data points that will be plotted on the chart. “**`true`**” specifies whether the chart resizes disproportionately when the window is resized (if it was “**`false`**”, the pie chart would always remain a circle). **`10.0`** specifies the distance between the distinct parts of the pie chart (`0.0` would mean that there would be no gaps between the parts).

The above code gives a result that looks like this:



As an example, this is a piece of code that changes some of the input parameters as discussed previously.

```
x=[50,50,75,75,100,100,125,125,150,150,175,175,200,200]

createPlot("pie")

pie.plotPie(x, false, 0.0)
```

INCOMPLETE

6.3.6.8 Bode, Step Response, and Root Locus Plots

Three types of transfer functions can be plotted: bode plot, step response plot and root locus plot. They all use polynomials in the numerator and denominator which can be specified by the user.

The step plot is used to display the output of a step response function. The bode plot plots the phase angle and amplitude of a function against its frequency. The root locus plot plots the root of a 2nd order differential equation on a complex plane.

Polynomials can be written in two formats.

```
n1 = #1, 0.1, 7.5#;

d1 = #1, 0.12, 9.0, 0, 0#;
```

The command above will create a polynomial n1 that equals $x^2 + 0.1x + 7.5$. The leftmost constant is the coefficient for the highest exponent of the polynomial. The polynomial d1 equals $x^4 + 0.12x^3 + 9x^2$.

```
n2 = polynomial([9]);

d2 = polynomial([1,2,9]);
```

The command above will create a polynomial n2 that equals 9. The polynomial d2 equals $x^2 + 2x + 9$.

To create a bode plot, the following command can be used.

```
createPlot("bode")

bode.plotBode(n2,d2)
```

This creates a plot window “bode” and plots a bode plot using n2 and d2. The first parameter taken by the plotBode function is the numerator and the second parameter taken is the denominator. Similarly, a step plot can also be plotted.

```
createPlot("step")

step.plotstep(n2,d2)
```

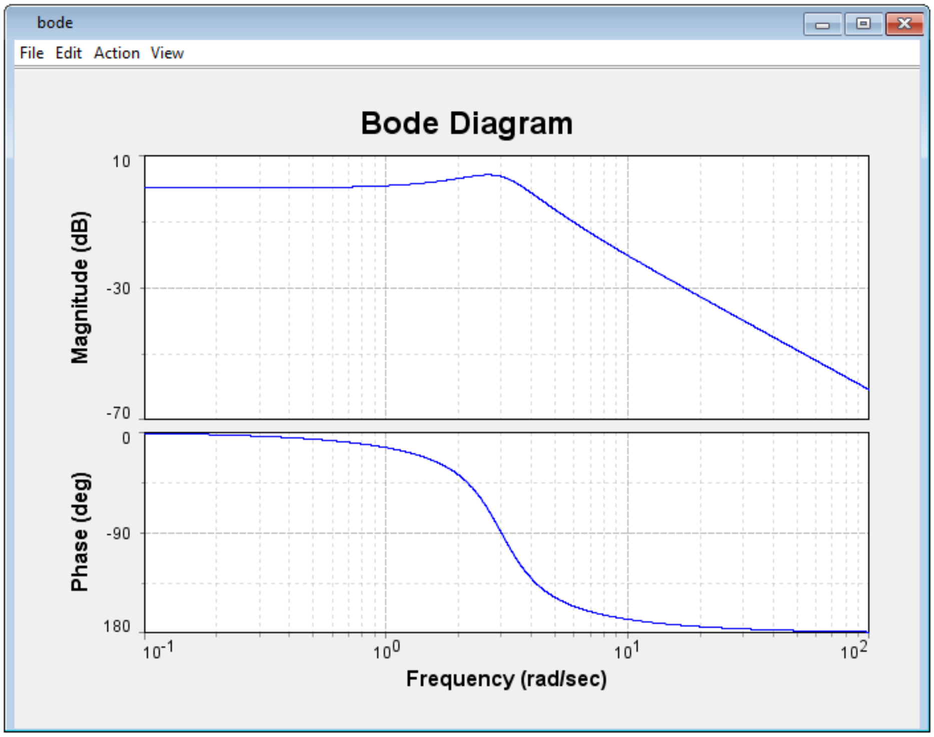
This creates a plot window “step” and plots a step plot using n2 and d2. The first parameter taken by the plotstep function is the numerator and the second parameter taken is the denominator.

A similar process is also used for a root locus plot:

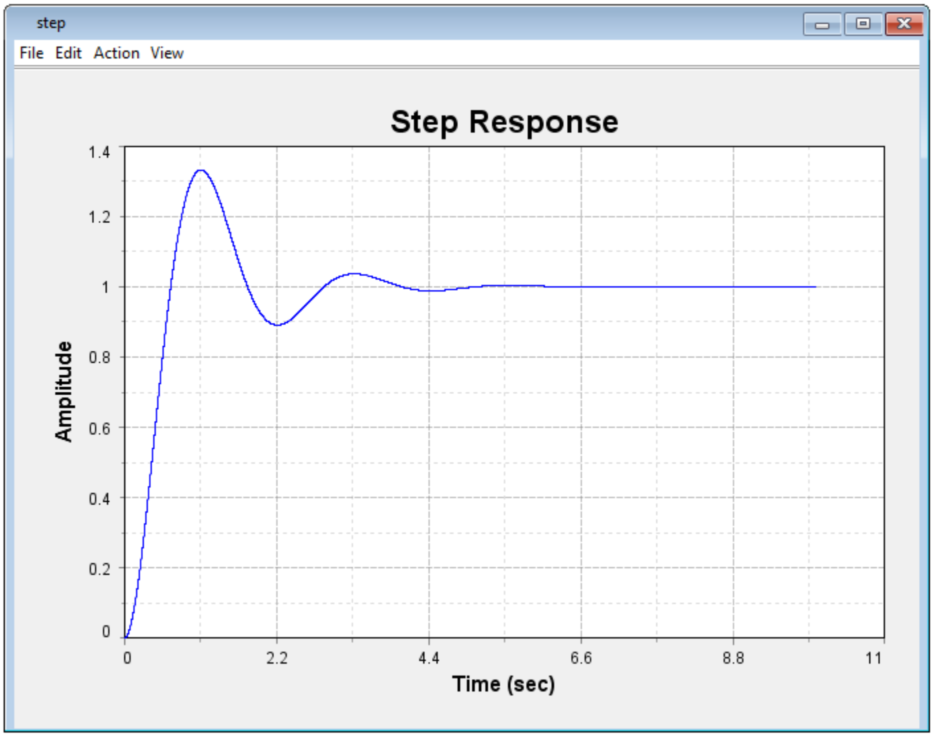
```
createPlot("PlotRL1")

PlotRL1.plotRootLocus(n1,d1)

PlotRL1.axis(-0.6,0.6,-6.0,6.0)
```

This creates a plot window “PlotRL1” and plots a root locus plot using n2 and d2. The first parameter taken by the plotRootLocus function is the numerator and the second parameter taken is the denominator. The additional third line is used to define the real and imaginary axes of the plot. The code follows the following pattern:

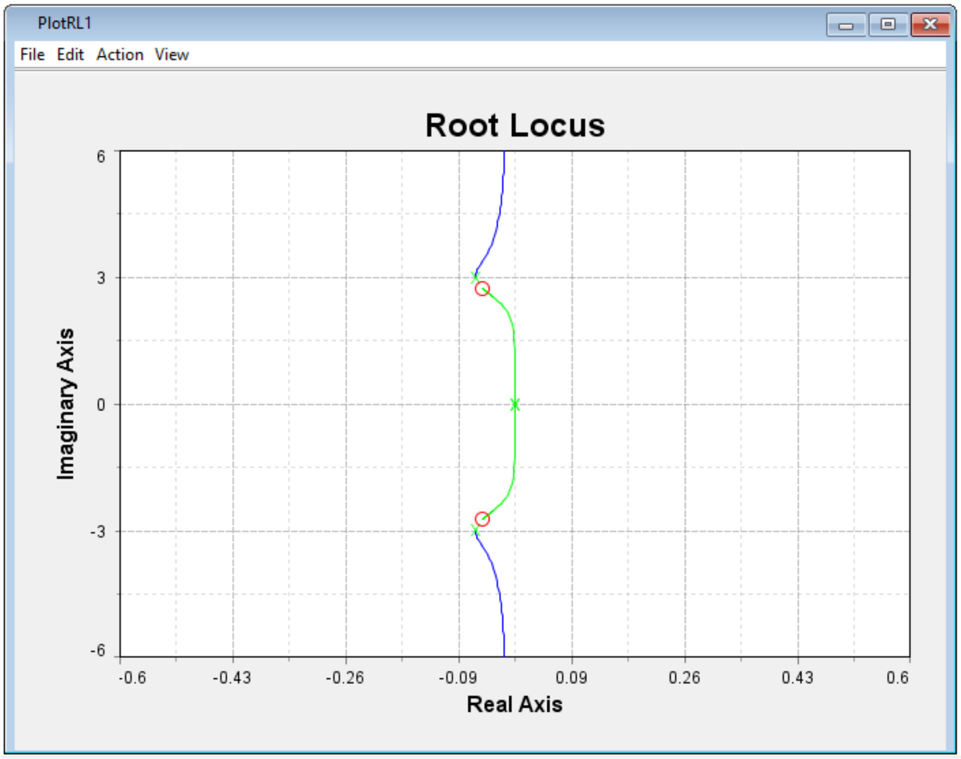
The results of all plots are shown below.



Bode Plot



Step Plot



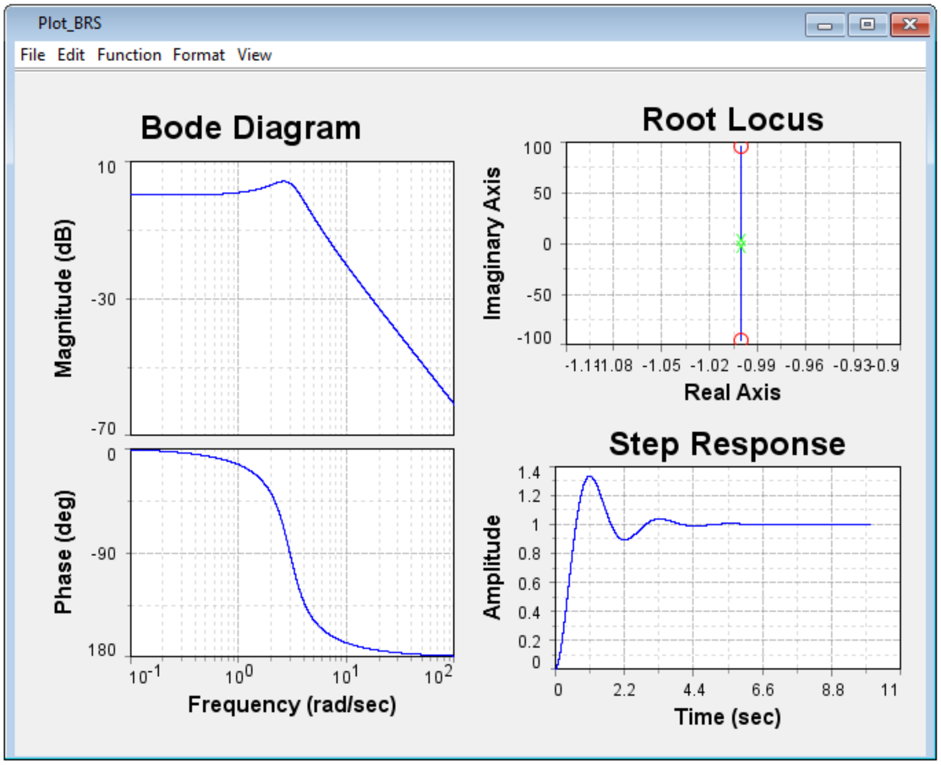
Root Locus Plot

6.3.6.8.1 Control Plot

There are two scripts in the scripts folder called “plotcontrol4” and “plotcontrol2x2”.

The plotcontrol4 creates four separate transfer function plots using the same data. The plotcontrol2x2 script creates a single plot window with the four plots in a 2 by 2 grid. The data points are created in the same way as the other examples, and the 2 by 2 grid is created like the previous example in 5.1.4.3.2.

The 2 by 2 grid looks like the image below.



6.3.6.9 3D Line Plot

A 3D line plot is used to plot data values on a three-dimensional axis. It is similar to the line plot except there is an added dimension.

An example of a 3D line plot data is given below:

```
t = 0.0:PI/5.0:10.0*PI

n = t.length

x = zeros(n)

y = zeros(n)

for i in 1:n

{

  x[i] = sin(t[i])*5.0

  y[i] = cos(t[i])*5.0

}
```

This particular example creates a helix on the axes x, y and t.

The line `t = 0.0:PI/5.0:10.0*PI` creates a list of datapoints for the t axis, starting at 0 and ending at 10*PI, with increments of PI/5 in between.

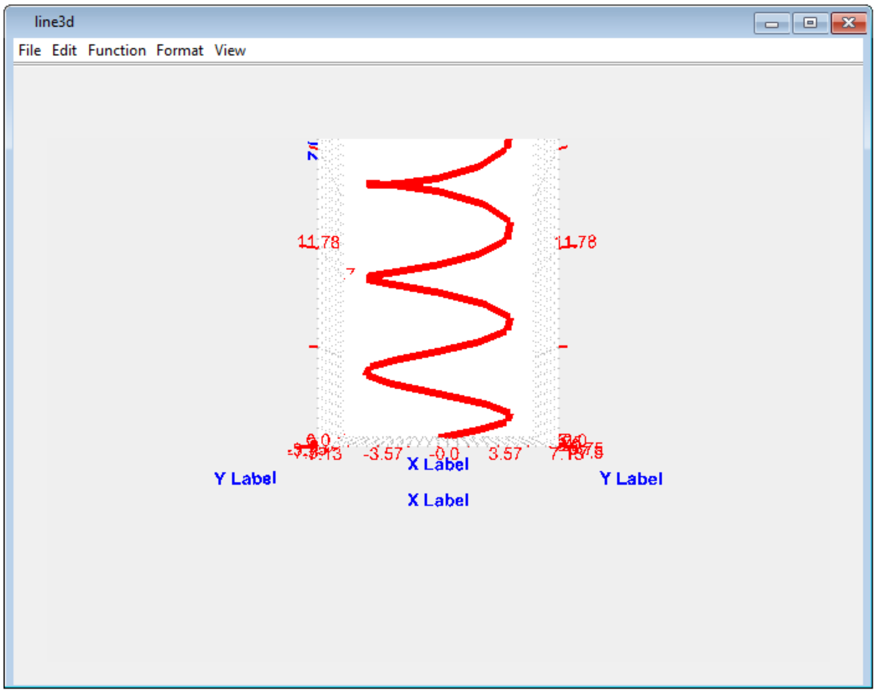
The line `n = t.length` creates a variable n with the value that is the number of data points in t. The lines that follow create lists of datapoints for the x and y axes, with a list of “n” zeroes.

The for loop that follows iterates through the numbers one through n, to populate the lists with the sine and cosine values. The sines of all values in t are written into x, and the cosines of all values in t are written into y.

```
createPlot("line3d")

line3d.plotLine3d(x,y,t)
```

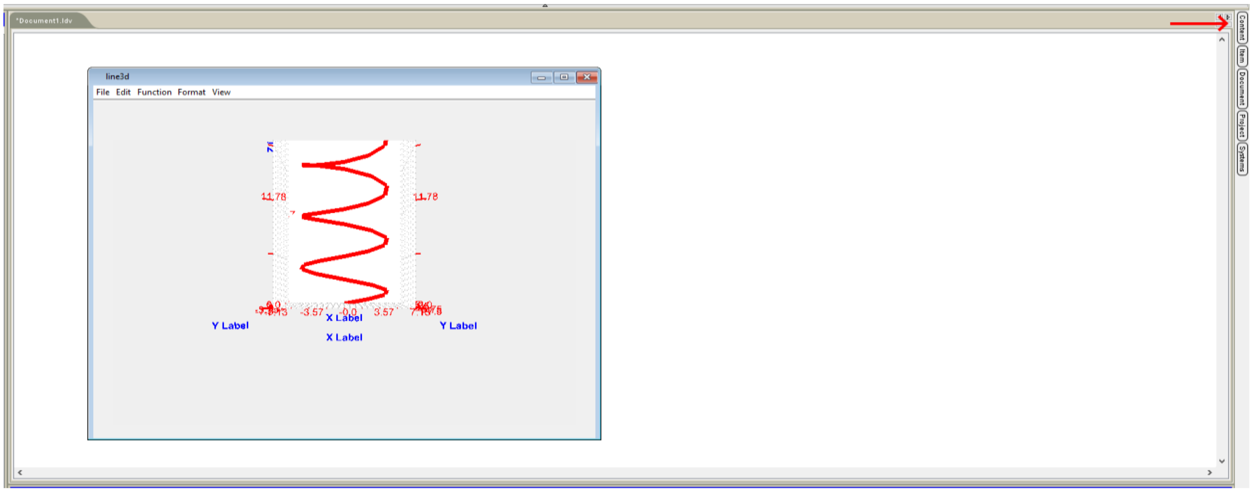
The two lines above are written after the data is created and are used to draw the plot onto the document. The first line creates a new plot named “line3d”, and the second line plots the data onto the plot called line3d. The parameters of the function are ordered as x, y and z axes. In this example, the data points in x are plotted on the x axis, y on the y axis, and t on the z axis.

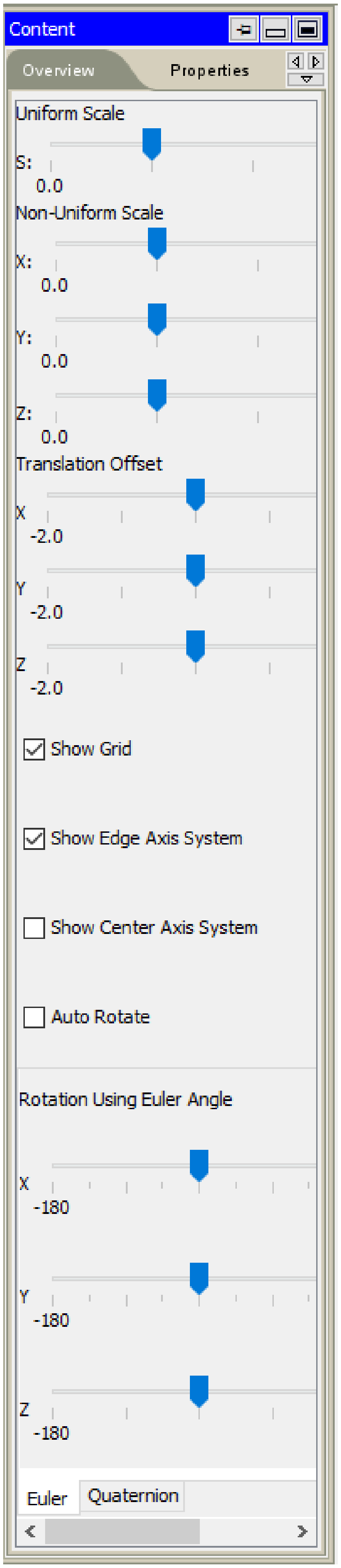
The resulting plot looks like the image below.



3D plots can be moved, rotated and scaled. Left clicking on the plot and moving the cursor will rotate the plot in that direction. Right clicking on the plot and moving the cursor will pan the plot in that direction.

Additionally, there are sliders to adjust the axis positions and scaling for the plot. They can be accessed by clicking on “Content” to the right of the Document Pane.





Uniform Scale

S: 0.0

Non-Uniform Scale

X: 0.0

Y: 0.0

Z: 0.0

Translation Offset

X -2.0

Y -2.0

Z -2.0

☑ Show Grid

☑ Show Edge Axis System

☐ Show Center Axis System

☐ Auto Rotate

Rotation Using Euler Angle

X -180

Y -180

Z -180

The Content tab looks like the image to the left.

The Uniform Scale slider uniformly scales the size of all the axes. The sliders in the Non-Uniform Scale section scale only the x, y and z axes respectively.

The sliders in the Translation Offset section pan across the axis that is labeled next to the slider.

The Show Grid checkbox shows the grid behind the plotted line when checked.

The Show Edge Axis System checkbox shows the x, y and z axes on the edge of the plot when checked.

The Show Center Axis System checkbox shows the x, y and z axes through the origin of the plot when checked,

The Auto Rotate checkbox automatically rotates the entire plot about the origin through the y axis at a slow pace when checked.

other sliders

6.3.6.10 3D Mesh and 3D Surf Plot

The 3D Mesh and Surface plots are used to plot data points on a three-dimensional axis. While in a line plot, the data is plotted as a line, the surface and mesh plots are used to draw the "surface" of the plot. While the surface plot shows a "solid" surface, the mesh plot divides the surface plot into unit grids.

An example to show 3D mesh and surface plots is given below:

```
x = -8:0.5:8;
y = x;
numOfPoints_x = x.length
numOfPoints_y = y.length
s = time();
z = zeros(numOfPoints_x,numOfPoints_y)
for i in 1:numOfPoints_x
{
   for j in 1:numOfPoints_y
   {
      R = sqrt(x[i]^2 + y[j]^2) + 1.0e-12;
      z[i,j] = sin(R)/(R*0.1) + 1.0;
   }
}
e = time();
d = e-s;
```

The above code can also be found on the "datasurf.hyp" file. The code is used to generate data for a Mexican hat function on the x, y and z axes using the x, y and z variables respectively.

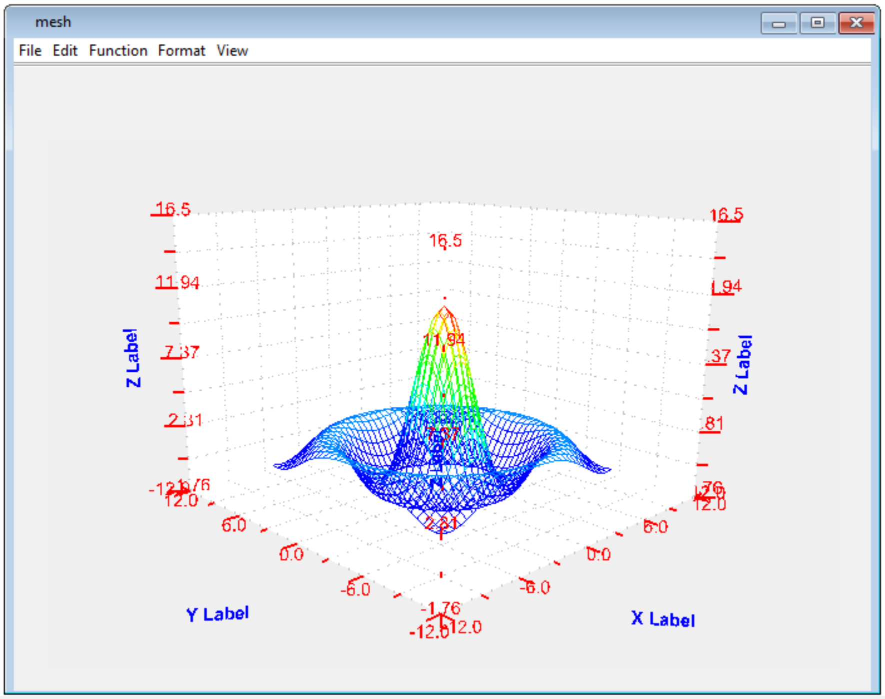
A mesh plot can be drawn using the code:

```
createPlot("mesh")
mesh.plotMesh(x, y, z)
```

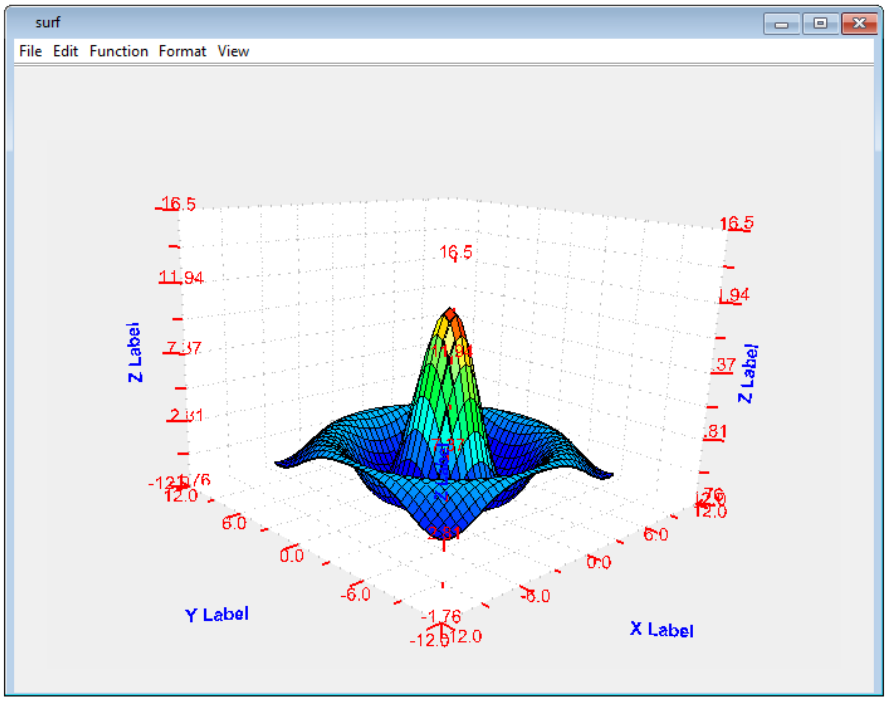
A surface plot can be drawn using the code:

```
createPlot("surf")
surf.plotSurf (x, y, z)
```

A 3D Mesh plot looks like the image below:



A 3D Surface plot looks like the image below:



Both plots can be scaled and rotated like the 3D line plot.

6.3.7 Console

The console is used to type and run commands, the results of which can be seen in the document pane or the console window.

6.3.7.1 Console Layout

6.3.7.1.1 Data notation

The console can display numerical answers in regular, scientific or engineering notation. To change this, click on the drop-down menu next to "Notation" and select your preferred notation.

The regular notation will simply display a decimal number. The scientific notation will display the number with a single digit decimal and exponents of ten. The engineering notation will display the number with a decimal and exponents of 10, where the exponents are multiples of 3.

6.3.7.1.2 Built in commands

The console has several built in commands to perform certain tasks. They can be found in the far right corner of the console pane by clicking on "Commands …" to get to a drop down menu.

`pwd` will display the path of the present working directory of the console. The working directory is the folder from which scripts can load by default without typing the entire path in the console.

`cd` will change the working directory to the parent folder of the original working directory and then display the path on the console.

`ls` will display the contents of the working directory on the console.

`ws` will display the present variables in the console. That includes any defined variables (see basic algebra below) and the default variables E and PI.

`wsd` will display the present variables and their data types (real, integer, imaginary, complex etc.)

`wsv` will display the present variables, their data types and their numerical values.

`wsf` will display all the available functions and their data types in parentheses.

`wsfd` will display all the available functions and their data types in a separate column.

`wsl`

`clear screen` will clear anything displayed on the console.

`clear var` will clear any created variables. Default variables will stay.

`clear fun` will clear any custom functions that have been created through scripts. Default library functions will stay.

`clear all` will clear everything including the display, variables and functions.

6.3.7.2 Console Functions

6.3.7.2.1 Basic Algebra

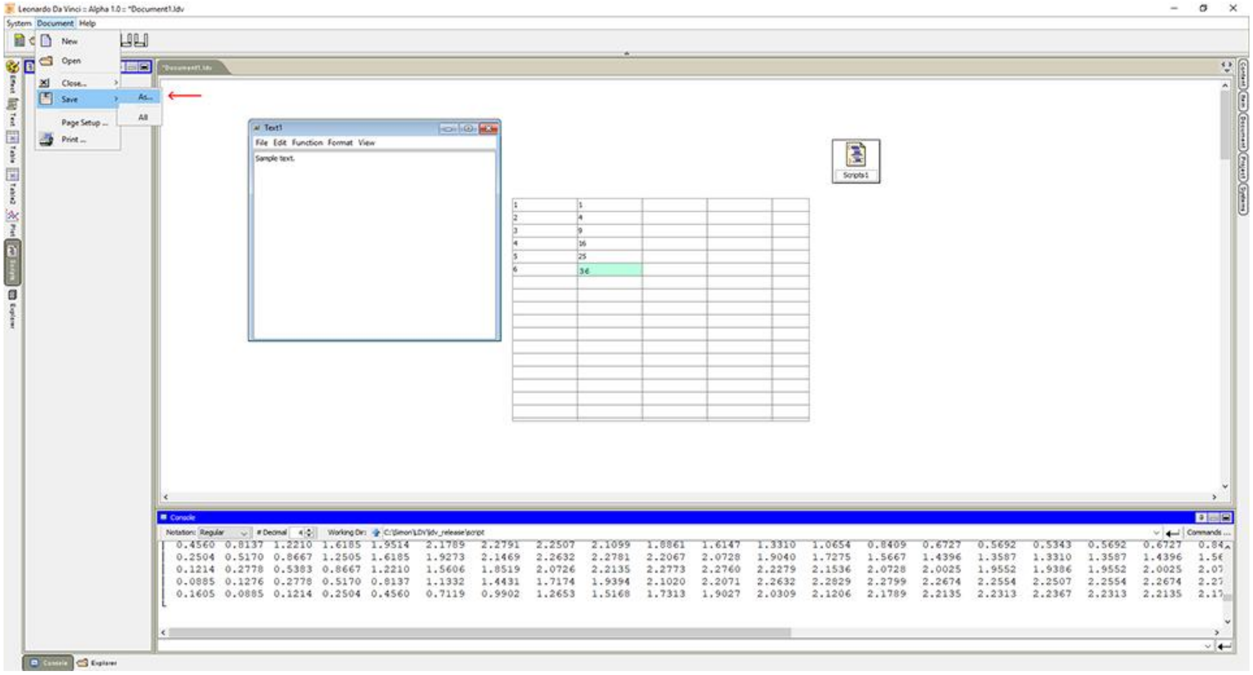
The console can be used to define variables and perform calculations using those variables. For example, to do a simple addition:

1. Type in a = 3. This defines a variable a and gives it a constant value of 3.
2. Type in b = 5. This defines a variable b and gives it a constant value of 5.
3. Type in c = a + b. This defines a variable c and makes it equal to the sum of a and b. The value of c will change if the values of a and b are also changed.

Similar calculations can be done to subtract, multiply, divide, create exponents etc. Reference for all of the available functions can be found in the Hyper Language Reference.

6.4 Saving a Document

To save a new document, go to Document > Save > As.. on the toolbar, as shown in the image below.



Sample text.

1	1
2	4
3	9
4	16
5	25
6	36

Notation: Regular # Decimal 4 Working Dir: C:\Simon\LDV\ldv_release\script

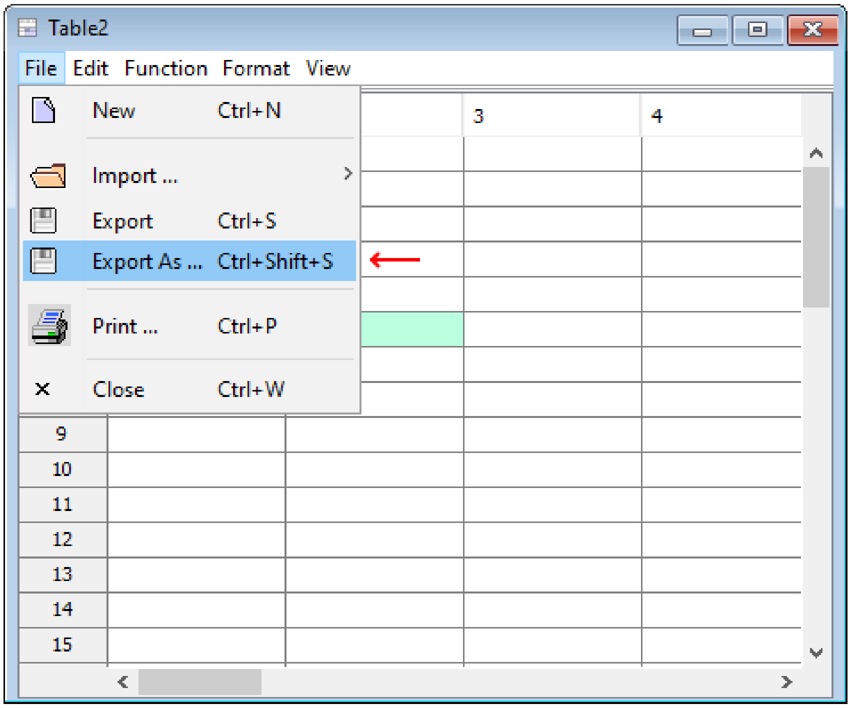
```
 0.4560 0.8137 1.2210 1.6185 1.9514  2.1789  2.2791  2.2507  2.1099  1.8861  1.6147  1.3310  1.0654  0.8409  0.6727  0.5692  0.5343  0.5692  0.6727  0.84
 0.2504 0.5170 0.8667 1.2505 1.6185  1.9273  2.1469  2.2632  2.2781  2.2067  2.0728  1.9040  1.7275  1.5667  1.4396  1.3587  1.3310  1.3587  1.4396  1.56
 0.1214 0.2778 0.5383 0.8667 1.2210  1.5606  1.8519  2.0726  2.2135  2.2773  2.2760  2.2279  2.1536  2.0728  2.0025  1.9552  1.9386  1.9552  2.0025  2.07
 0.0885 0.1276 0.2778 0.5170 0.8137  1.1332  1.4431  1.7174  1.9394  2.1020  2.2071  2.2632  2.2829  2.2799  2.2674  2.2554  2.2507  2.2554  2.2674  2.27
 0.1605 0.0885 0.1214 0.2504 0.4560  0.7119  0.9902  1.2653  1.5168  1.7313  1.9027  2.0309  2.1206  2.1789  2.2135  2.2313  2.2367  2.2313  2.2135  2.17
```

That will show the save window (in the form of a file browser), and the file can be saved in the preferred location with the file format .ldv. The .ldv file will contain all texts, tables, scripts etc. created within the document.

To save an existing document, click on the Save icon.

6.4.1 Saving app documents

The texts, tables and scripts created within the document can be saved separately and then imported into different documents. To do so, go to File > Export As… on the internal document that is being saved. (Alternatively, the keyboard shortcut Ctrl + Shift + S can also be used)



That will show the save window (in the form of a file browser), and the file can be saved in the preferred location. Texts will be saved in .text format, tables will be saved in .table format, plots will be saved in .plot format, and scripts will be saved in .hyp format.