Laplace

P!\K/AVOP|Un0X

COLLABORATORS

	TITLE : Laplace	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	P!\K/AVOP|Un0X	June 8, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Laplace

1.1 Laplace ©1996 by A Vision of Paradise

Welcome to Laplace V0.3 (Evaluation)

Getting started :

* What is Laplace ?
* Legal stuff
* Registration
* That's me !!!
* Greetinx go around the world..
* Requirements
* Installation
* Still left to implement..
* Done so far..
* Shit happens..

Usage :

* Working with Laplace
* Projects
* Edit capabilities
* Menu description
* Available object types
* Defining objects
* Available functions
* ARexx port
* Tooltypes
* Presets
* Libraries
* Init.lh
* math.lh
* physic.lh

1.2 Introduction

Introduction :

Laplace is (or at least should be) an universal tool for mathematical calculations. It lacks such a GUI with hundreds of windows like those other progs I've found on AMIGA. Instead it works like a shell, you enter a command and get the result displayed, which is much more flexible.

I started programming Laplace, because I was looking for a program for handling matrices, what I needed for my studies. First all programs I found were only able to handle floatpoints, but especially in matrix calculations this leads quite fast to large errors. So I made a simple prog that handled fractions instead of floatpoints. After that I needed something to handle parameters in matrices, and so I started totally rewrote Laplace. This is the result.

Currently I know only one program on AMIGA that is able to do things like this : MAPLE V. Well, it is millions of lightyear better that Laplace, but feature has it's price : DM1500!!!!! You cannot compare Laplace and MAPLE, neither in power nor price...

Currently Laplace is everything else than complete. As you notice, even the icon promisses more than Laplace can keep, because Laplace is not even possible to calculate simple things like derivations ;-)

1.3 Copyright

Copyright :

This is only an evaluation version of Laplace. I plan to make it SHAREWARE, when it reaches a reasonable state. For know Laplace is FREEWARE.

You may distribute Laplace only as the original archive. You may uuencode or ??? it, but you must not change the contents.

You are not allowed to sell it or take more than $4, DM5.- etc for copying it.

Use it on your own risk. If you plan to build a nuclear plant, I take absolutely NO responsibility for uncontrolled chain reaction caused by false calculations!!!

Laplace uses MUI V3.3 by Stefan Stuntz.

Laplace was compiled with DICE V2.06 by Matthew Dillon.

Laplace was debug with Enforcer V37 by Michael Sinz.

Texts were edited with GoldED V3.1.3 by Dietmar Eilert.

1.4 Registration

Registration :

Since Laplace is currently FREEWARE, there's no need to register. But you may send comments, bug report, flames etc. of course. Send it to...

1.5 Author's address

The author :

This master piece of programming arts was done by

P!\K of /|_ __ . /\ |__: /|____ /__:__/\

/ | \ :/ \ | |/ \/ | \

_/ | _| \ \| _/ |\ ___ ___/

\ | | \ \ \ |/ / \

\ _____|____\ _/\ _____/\ __|__ /

\| : \/ \| \/ · \/

s-mail :

Benjamin Stegemann

Rohrbacher Str. 66

69115 Heidelberg

Germany

e-mail :

bstegema@ix.urz.uni-heidelberg.de

bstegema@urz-mail.urz.uni-heidelberg.de

Actual versions of Laplace can be found in the Aminet directory misc/math.

If you are interested in other programs I've done so far, look at these :

ShuffleRun

Funny arcarde game for two or more players, including tournament mode.

TankHunter

Two-player tank blasting...

Writer

Creates totally stupid texts, based on any ASCII input. Even greatest literature will be shamelessly blown up!!

TreeGrow

Create fractal tree images.

These progs are not available on aminet, sorry, but it's quite old stuff..

Oh, almost forgot it, they are published on Fish disks, wait a moment, I will see which ones..

Jo, back again, it's 814 (TreeGrow), 815 (ShuffleRun) and 818 (TankHunter). Writer is (or was) available on BBS.

If you are really interested, send me a mail.

1.6 Greetinx

Greetinx :

Thanks must go to :

Stefan Stuntz for is fantastic MUI.

Matthew Dillon for DICE.

Dietmar Eilert for GoldED.

Martin Huttenloher for MagicWB.

Mattias p. Eriksson for additional icons from is MagicRabbit collection.

Kai Iske for MagicCX.

Stefan Sommerfeld and Michael Knoke for MCP.

Michael Sinz for Enforcer.

And all other folks around this crazy planet, who support the AMIGA !!!!

P!\K of /|_ __ . /\ |__: /|____ /__:__/\ . >P!\K/The Un0X Project<

/ | \ :/ \ | |/ \/ | \ :

_/ | _| \ \| _/ |\ ___ ___/ | A2000/3.5MB

\ | | \ \ \ |/ / \ | A1200/030/6MB

\ _____|____\ _/\ _____/\ __|__ / :

\| : \/ \| \/ · \/ .

1.7 Requirements

Requirements :

OS3.0 (V38) or better.

MUI3.3 which may be found in aminet : dev/gui/muiXXusr.lha.

avop.library supplied in this archive.

and a FAAAAAST AMIGA...

1.8 Installation

Installation :

Since Laplace is still under development, I didn't write an installer script ;-)

But, don't be afraid, installing is really easy!! Just drag the whole Laplace directory to any place you like, no libs, fonts, catalogs to install. Even an assign isn't needed anymore.

You may add the assign "Laplace:" to the Laplace directory, then this assign will be uses as PROGDIR:, but this is currently not very useful at all (except for me, as I need this to find all files whenn debuging...).

1.9 Working with Laplace

Working with Laplace :

Laplace works much like a shell, you enter an expression press return and Laplace gives you the result (after some time ;-). See editing or menus for more details.

Laplace's processing is based on objects. You can define new objects and reference them later.

When a new window is opened, Laplace will do several things to setup the environment:

1) load options from Presets/Default.

2) load F-keys from Presets/Default.fkey.

3) execute the command include(Init.lh). See also Init.lh.

You may modify all these files, to adjust the environment to your needs.

You can load and save sessions, see Projects.

1.10 Editing

Editing :

Well, I think the basic edit functions are now available.

Clipboard is supported, but currently only whole lines can be cut, copied and pasted.

Insert line inserts a line before the current line, to insert a line at the end, just press return in the last line.

Double clicking on a line, works just line pressing return.

Here are the keys that you can use for editing :

Key: Description:

backspace

del

left

right

up

down guess what..

shift left

shift right advance one word

alt left goto beginning of line.

alt right goto end of line.

alt up goto first line.

alt down goto last line.

ctrl up move one item up in history.

ctrl down move one item down in history.

ctrl-shift up move to first item in history.

ctrl-shift down move to last item in history.

esc undo changes in current line. This work

only, as long as you don't leave

the line.

return evaluate the current line and advance

to the next line.

shift return evaluate the current line and insert a new

line after it.

alt return split the current line at the cursor position

and insert a new line after it.

shift backspace erase current line from cursor position

to the beginning of the line.

shift del erase current line from cursor position

to the end of the line.

alt backspace clear the current line.

alt del clear the result of the current line.

ctrl tab pop up a list of all available objectes

of the current context. Select an entry

and it will be inserted at the current

cursor position.

alt tab pop up a list of all available functions.

Fn

shift Fn insert the string linked to a F-key.

1.11 Menu

Menu :

Project :

About MUI

MUI, what else ;-)

About

;-)

New window

Open a new, blank window.

Clear

Clears the current window.

Load..

Open a project file.

Save

Save a project file. Use the current path or select a new.

Save as..

Save a project file. Select a new path.

MUI Prefs

Open MUI preferences.

Quit

NOOOOOOOO :-O

Edit :

Cut

Copy

Paste

Insert line

Remove line

see editing.

Windows :

Function list

Opens a window with all available functions. A doubleclick on an entry will insert the function into the current line.

Debug

Opens the debug window with a list of all debug strings.

Options :

Options are specific for a project, they are saved in the project file.

Always use float

Use always floatpoints even if fractions could be used.

Simplify

Enable simplification. If Disabled calculation is a bit faster, but the result look quite ugly.

Precision...

Laplace tries to convert floatpoints into fractions whenever possible. But the floatpoint routines are not too exact, so there might be an small difference to the correct value (e.g. 1.2000000000000001 when it should be 1.2). If the difference is smaller than the specified precision, Laplace assumes the value as a fraction and converts it (in this case to 6/5). This is not applied to value near zero, so 1e-30 won't be converted to 0.

F-Keys...

Define strings linked to the F-keys. The specified text will be inserted into the current line, if you you hit an F-key.

Load F-Keys

Save F-Keys

Load or save a set of F-keys. You may load F-keys from other project files.

Load options

Save options

Load or save the state of Alway use float, Simplify and Precision. You may load options from other project files.

Help :

Index

Loads this file, displaying the main page.

Functions

Loads this file, displaying this page.

Search...

Opens a window, where you can specify a keyword to search for. If the keyword is found the file will be opened on the correct page. Note that text window runs asyncron, which means that you can leave the window opened while working. You don't even have to close the window to continue the search. Just bring the 'Searching...' window to front, select 'Continue' and the next page containing the keyword will be loaded into the textwindow.

1.12 Projects

Projects :

Laplace saves project files as plain ASCII files, you may edit them with any standart texteditor. Projects are usually placed in the Projects directory.

All options (see menu) are save in the project file, the format as in preset files is used.

1.13 Presets

Presets :

Options are saved as plain ASCII files, and are usually stored in the Presets directory. The can be edited by any standart texteditor.

The line format is quite simple: a line starts with an # followed by a keyword, a = and the value for this option. You must not insert any spaces, except otherwise stated below. E.g.

#USEFLOAT=TRUE

#ERRBND=12

Options #?.opts

These files contain the values for Alway use float, Simplify and Precision. The keywords are :

#USEFLOAT - set to TRUE or FALSE.

#SIMPLIFY - set to TRUE or FALSE.

#ERRBND - an integer value between 1 and 32.

F-keys #?.fkey

These files contain the definitions for the F-keys. The rest of the line will be used as the string, spaces are allowed.

#F1...#F10 - normal F-keys 1..10.

#F11...#F20 - shifted F-keys 1..10.

1.14 Expressions

Expressions :

Expressions are entered in a usual manner, e.g.

1+2/3*(4-2)<RETURN>

Guess what's the result... More formally, an correct expression is :

a or

a op b

where is +, -, *, / or ^ (power of) and

a,b are valid

objects,

references,

functions,

expression lists or

expressions.

You may enter several expression at once, seperated by commas, e.g.

1+2,2/4,12+10

The results will be displayed seperated by lines.

1.15 Expression list

Expression list :

An expression list is an list of expression seperated by commas embedded in {} brackets or supplied to the do() command.

All expressions will be evaluated, but only the last result will be used. The result of the list will be this last result. This makes only sense with functions like debug.

E.g.

sin({debug("Hello world!"), pi/2})

1.16 Types

Types :

Currently support currently four types of objects : Real, Vector, Matrix and Equation

Real :

These are just numbers as well all know them (or do we know them all ?!).

Numbers can be entered in exponential style, which is {+/-/<nothing>}ddddd[.ddddd][e{+/-/<nothing>}dddd]. Ufff, but it's quite easy ;-) to enter e.g. 6.626 * 10^(-34) write 6.626e-34. The (whole) number after 'e' is just the exponent. Don't use brackets, if the exponent is negative!

As long as you enter only whole numbers, Laplace tries to use fractions, so enter 1/2 instead of 0.5. This way you can enter things like 1/7 exactly, and you won't get result which are almost exact zero, when it should be exact ;-)

Vector :

A vector is a tupel of reals. To create a vector (2,4,1) enter vector(2,4,1) or as a short form [2,4,1]. A vector may of course contain references to other (real) objects.

You can add and multiply (scalar) two vectors, or multiply a real or matrix with a vector (in this order, not vector * real!).

Matrix :

A matrix is something like a 2-dimensional vector.. (if you don't know what's this, you probably won't need them ;-) A matrix is created by e.g. matrix([1,a,3],[5,7,b],[c,3,4]) or as a short form [[1,a,3],[5,7,b],[c,3,4]] where [..] is a column of the matrix. All columns must be of the same height of course.

You can add and multiply two matrices, or multiply a real and a matrix (in this order, not matrix * real!).

Equation :

To create a equation use the eq command.

You can of cource calculate with equations. The operations will be applied on both side of the equation. You can add or multiply two equations. If you add or multiply a real, vector or matrix to an equation, the equation has to be on the left side, e.g.

a=eq(2*x+4,0)

a=a-4

a=a/2

You can also apply functions line sin(), exp()... on equations, e.g.

a=eq(ln(x),12*e)

a=exp(a)

1.17 Definitions

Definitions :

If you want to use an result later or just define a variable, write name = expression. This creates an objects that can be referenced by it's name in later (!) lines. If you reference to an object, Laplace searches from the actual line back, so you may define an object several times, but only the latest version is used. E.g.

[1] a=1/2

[2] 2*a

gets 'a' from [1]

[3] a=a+1

'a' on the right side references to the definition in [1]

and creates a new object called 'a' to be referenced below

[4] a

gets 'a' from [3]

If you move back to [2], you still get the same result, because the new 'a' is defined below and cannot be reference from [2].

There are some special conventions for object names. Valid characters are A..Z Ö Ä Ü a..z ö ä ü 0..9 @ § $ '. You may not use a number as the first character.

There are two special characters which may be used : ~ (tilde) and _ (underscore). _ as the first character will create a line above the name, which is often used in mathematics. Inside the name _ will create an index, character after _ will be used as the index at the bottom of the name. ~ work similar, but creates an index at the top. You can use both indexes. E.g. _a, a_1, a~1 or everything at once _a~2_1.

Starting with version 0.3 Laplace support the usage of greek symbols. If the name or an index matches the name of a greek letter, then this letter is used to display the variable. Number may follow the letter name. E.g. alpha1_beta. A name in lower case represents a small greek letter; if the first character of the name is uppercase, then you get an big greek letter. All supported names are :

alpha, beta, gamma, delta, epsilon, zeta, eta, theta, jota, kappa, lambda, my, ny, xi, omikron, pi, rho, sigma, tau, ypsilon, phi, chi, psi, omega,

Alpha, Beta, Gamma, Delta, Epsilon, Zeta, Eta, Theta, Jota, Kappa, Lambda, My, Ny, Xi, Omikron, Pi, Rho, Sigma, Tau, Ypsilon, Phi, Chi, Psi, Omega

There are three difference kinds of definitions : variables, parameters and constants.

When you reference a variable, it's contents will be inserted. Variable are defined using name = expression. E.g.

a=3

a+1 -> 4

A reference to a parameter will not be evaluated, it will remain in the result. Use name := expression to create a parameter. If you want to get the final result use the eval() function. E.g.

a:=3

2*(a+2) -> 2*a+4

eval(2*(a+2)) -> 10

Constants have no value, so they won't be evaluate, even with the eval() function. They are defined using the const() function. E.g.

const(a)

2*(a+2) -> 2*a+4

eval(2*(a+2)) -> 2*a+4

You can also define functions. Just enter a parameter list embedded in bracket after the object name, e.g.

f(x):=x^3

The parameters will be handled like constants in the function expression. If you want to use other object type than reals, you have to specify the type in the parameter list, the format is line in const().

When referencing a function you have to specify the parameters to be inserted into the function's expression, e.g.

f(3) -> 27

const(a)

f(a+1) -> (a+1)^3

1.18 Tooltypes

Tooltypes :

You may enter some tooltypes to the icon of laplace. To do this, select the icon on the workbench and choose Icons/Information from the workbench menu. Then modify the list of tooltypes using the (quite bad) edit functions.

Tooltypes are supplies using this format :

KEYWORD=value

Unknown keyword will be ignored, syntax is caseinsensitive.

Supported tooltypes are :

AREXXONLY (boolean)

Set to YES or NO. If set to yes, Laplace won't open a window on startup, but starts iconified; only an application icon will be shown on the workbench. The first window is opened when doubleclicking on the application icon or uniconifying Laplace on another way. This is useful, if you want don't want to use Laplace directly, but only through it's ARexx port.

..not much eh..

1.19 Libraries

Libraries :

Laplace offers the ability to process external files. These files are plain ASCII files and may be edited with every texteditor. These files will be processed, just as if each line would be entered, blank lines and lines with leading ; will be ignored. The results won't be displayed, so this is only useful for constant and function definitions. Library files have a trailing .lh and are located in the Include directory. Use the include() command to gain access to a library.

Some basic files are included :

Init.lh

math.lh

physic.lh

If you want a library to be available on startup add a line source(libname.lh) to the file Include/Init.lh, which is always processed on startup.

1.20 Init.lh

Init.lh :

This file will always be processed on startup. Any definitions or commands that should always be done can be placed here.

1.21 math.lh

math.lh :

some useful mathematical definitions

pi:=3.141592653589793 ;-)

e:=2.718281828459045 :->

err(x):=exp(-x^2) error function

1.22 physic.lh

physic.lh :

some useful physical definitions

G:=6.67e-11 gravity constant

g:=9.81 earth acceleration

N_A:=6.0225e23 avogadro number

R:=8.31 gas constant

k:=eval(R/N_A)

m_p:=1.6725e-27 a proton's mass in kg

m_n:=1.6747e-27 a neutron's mass in kg

m_e:=0.911e-30 an electron's mass in kg

u:=1.66042e-27 nuclear mass unit in kg (1/12 of C12 atom nucleus)

h:=6.626e-34 planck constant

_h:=eval(h/(2*pi)) h/(2*pi)

m_sun:=1.99e30 the sun's mass in kg

r_sun:=6.96e8 the sun's radius in m

m_earth:=5.98e24 the earth's mass in kg

r_earth:=6.37e6 the earth's radius in m

d_earth:=1.49e11 the earth orbits's radius in m

m_moon:=7.34e22 the moon's mass in kg

r_moon:=1.74e6 the moon's radius in m

d_moon:=3.84e8 the moon orbits's radius in m

1.23 To do

To do :

As I said before, this is only an evaluation version, so there are quiet a lot of things left to do, before I release Laplace as SHAREWARE. If you have any suggestions, you are invited to send me a note!!

Here is a list of some major things :

- editing is not very comfortable now, improve it (history, block editing..)

- use CGFonts for output (is there anyone out there knowing how to work with the bullet.library?)

- support complex numbers

- function plotting, perhaps internal or external using ARexx (seems that Graph2D needs a better ARexx port..)

- export expressions as LaTeX,...

- calculate integrals (do you REALLY believe, I manage this...)

- a whole bunch of small improvementes...

- the greek font is rendered by IntelliFont and doesn't look very pretty (does any have a designed one ??)

1.24 Bugs

Known bugs :

If you find a bug, not documented feature etc. don't hesitate, TELL ME ABOUT IT!!

Here are those nasty things I know, but didn't manage to remove 'em (for now):

- my MUI-CustomClass 'eats' all keyboard input, disabling the tab-key and button shortcuts.. Currently there is only a workaround for the tab-key, but things are still quite ugly...

- you get sometimes problems with object names ending with an 'e', try m_e+pi (no spaces !!). m_e + pi (with spaces !!) works...

- I'm using the mathieeedoub*.library for floatpoint processing. Seems that it's not too precise; 0.4 * 5.0 is 2 $\pm$ 1e-16 !! When trying to convert floats into fraction, Laplace ignore differences smaller than 1e-15...

- Taylor approximation crashes, if you try to evaluate at an a, where the function is not defined, e.g. taylor((ln(x))(x), 0, 2).