Introduction to wxMaxima for Scientific Computations

Book · July 2018

1 author:

M. Kanagasabapathy
Rajapalayam Rajus' College

Introduction to wxMaxima for Scientific Computations

By
Dr. M Kanagasabapathy

BPB PUBLICATIONS
20 Ansari Road, Daryaganj, New Delhi-110002

FIRST EDITION 2018

ISBN: 978-93-8728-442-5

Distributors:

BPB PUBLICATIONS
20, Ansari Road, Darya Ganj
New Delhi-110002
Ph: 23254990/23254991

BPB BOOK CENTRE
376 Old Lajpat Rai Market,
Delhi-110006
Ph: 23861747

COMPUTER BOOK CENTRE
12, Shrungar Shopping Centre,
M.G.Road, Bengaluru–560001
Ph: 25587923/25584641

DECCAN AGENCIES
4-3-329, Bank Street,
Hyderabad-500195
Ph: 24756967/24756400

MICRO MEDIA
Shop No. 5, Mahendra Chambers,
150 DN Rd. Next to Capital Cinema,
V.T. (C.S.T.) Station, MUMBAI-400 001
Ph: 22078296/22078297

Published by Manish Jain for BPB Publications, 20, Ansari Road, Darya Ganj, New Delhi-110002 and Printed at Repro India Pvt Ltd, Mumbai

PREFACE

It is with great pleasure that I write this preface. The continued growth of the Maxima user community is very encouraging to me as a developer. I am proud to say that Maxima has continued to grow as a project over the years that I have been involved, and several related projects have been successfully developed, including the wxMaxima user interface. It is gratifying to see that Maxima and wxMaxima are useful to many people around the world, and with the support of the world-wide user community I am inspired to seek continued success with the Maxima project.

I first became involved with the Maxima project in 2003, when I was hoping to calculate some integrals related to the branch of statistics called survival analysis. I began to participate in the Maxima mailing list, and my first contribution to the code was the numericalio package to read and write data files. At some point I began creating releases, beginning with Maxima 5.9.3 or was it 5.9.2? which I continued for several years. I fixed many bugs, and addressed many messages to the mailing list. Through it all I have always enjoyed the easy-going camaraderie of my fellow developers. I have never met any of them in person, yet I feel that I know them well from our daily interactions.

I have often conceptualized the Maxima project as a sort of virtual workshop. The workshop has been standing for many years, and there are still a few old-timers around, although many workers have come and gone over the decades. The workers labor mostly on their own projects, as they decide are useful and appropriate, using the tools which are there in the workshop for anyone to use. Before embarking on a new project, often a worker will raise the issue and discuss different approaches with others on the mailing list. A discussion usually brings some consensus or general agreement about a task, before a worker begins to work on it. Or there may be no discussion. Tasks which are minor may be carried out without discussion; there may be discussion only after the fact, and sometimes even disagreement, leading to some work being undone. This too, is part of the overall picture of loosely-organized efforts in the workshop. We are all marching in more or less the same direction, but sometimes there are disagreements as to exactly where to go.I have to say that I find such an atmosphere very conducive to doing good work, and I believe this is a good model for software development in general.

Fundamental functions are methodically organized in this book in an articulate style, which might be absolutely useful for the beginners to understand the basics of Maxima. I hope that the readers of this fine contribution by Dr. M Kanagasabapathy will be inspired and may join us in the forthcoming Maxima project workshops. There is always more work to be done, and perhaps in times to come, future users of Maxima will thank the efforts of this author.

Robert Dodier
Developer & Project Administrator of Maxima
Portland, Oregon, USA

Author's Note

Scientific computing is an indispensable technique for physical, chemical and biological sciences or even for financial estimations to simulate the technical data through mathematical models. It plays a vital role from research and development to industrial processes optimization. Though many packages are available for numerical computations, very few packages are available for symbolic computations like Mathematica, Maple, MATLAB, FriCAS, Sage, Scilab, Axiom, Euler, SymPy, etc. Among these, Maxima is an open sourced free ware and has user friendly interface, yet capable of executing persuasive symbolic as well as numerical computations.

Though Maxima has copious, built-in computational and graphical functions with steep learning curve, this book is a beginner's guide and outlines the fundamental functions and algorithm of coding with simple examples. Basic functions are arranged alphabetically for quick reference.

I am sincerely indebted to **Dr. Robert Dodier**, USA, Developer & Project Administrator of Maxima and **Dr. Roland Salz**, Bochum, Maxima Module Developer for peer-reviewing the book manuscript, and for their suggestions despite their busy schedule.

I express my gratitude to our President, Secretary and college governing council members, Rajapalayam Rajus' College for granting consent to publish this book.

I am grateful to **Dr. V. Venkatraman**, Principal, Rajapalayam Rajus' College for his motivation to publish this work.

I extend my sincere appreciations to **BPB Publications,** India for formatting my work into book.

Table of Contents

(%o1) 0.2757205647717832

---

cspline (list)

returns the cubic spline interpolation polynomial for the given list.

To use this function first call, **load(interpol)$.**

(%i1) load(interpol)$ /* output suppressed */

(%i2) a:[[0, 1], [1, 2], [2, 33], [3, 244]]$ /* [x, y] list */

(%i3) cspline(a);

(%o3) $(-4x^3 + 5x + 1)\text{charfun}\,2(x, -\infty, 1) + (-46x^3 + 414x^2 - 985x + 715)\text{charfun}\,2(x, 2, \infty) + (50x^3 - 162x^2 + 167x - 53)\text{charfun}\,2(x, 1,2)$

/* find 'y' by interpolation from 'x' values */

/* Conditions

$(-4x^3 + 5x + 1)$	$x_1 \leq x \leq x_2$.	$(0 \leq x \leq 1)$
$(50x^3 - 162x^2 + 167x - 53)$	$x_2 \leq x \leq x_3$	$(1 \leq x \leq 2)$
$(-46x^3 + 414x^2 - 985x + 715)$	$x_3 \leq x \leq x_4$	$(2 \leq x \leq 3)$

For more details on cubic spline refer: Basics of Mathematical modeling for Science & Engineering Numerical Data Analysis, Dr. M Kanagasabapathy, Tech-Center, Amazon Co., (USA) 2013 I Edition, ISBN: 978-1492983125. */

---

cv (list) or cv (matrix)

returns the variation coefficient from the standard deviation and the mean for the given list or matrix. /* $cv = \frac{\text{standard deviation}}{\text{average}}$ */

To use this function first call, **load (descriptive)$.**

(%i1) load (descriptive)$

(%i2) a:[15.6, 17.5, 36.6, 43.8, 58.2, 61.6, 65.2, 72.6, 98.9];

(a) [15.6, 17.5, 36.6, 43.8, 58.2, 61.6, 65.2, 72.6, 98.9];

(%i3) cv (a), numer;

(%o3) $\begin{bmatrix} a & 2a & b & 2b \\ 3a & 4a & 3b & 4b \\ c & 2c & d & 2d \\ 3c & 4c & 3d & 4d \end{bmatrix}$

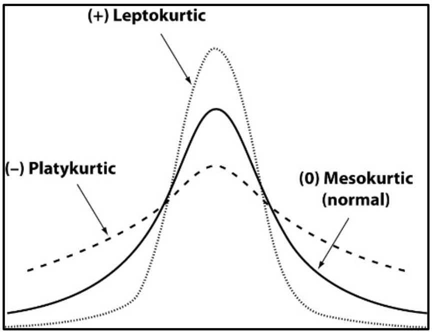
(%i4) kronecker_product (y, x);

(%o4) $\begin{bmatrix} a & b & 2a & 2b \\ c & d & 2c & 2d \\ 3a & 3b & 4a & 4b \\ 3c & 3d & 4c & 4d \end{bmatrix}$

---

kurtosis ([list])

returns kurtosis coefficient, for the list of data (for x and y set). To use this function first call, **load (descriptive)$**. It measures the sharpness of the peak of a frequency-distribution curve.



$k = \frac{\sum_{i=1}^{n} \frac{(x_i - \overline{x})}{n}}{n\, s^4} - 3$; s – standard deviation; $\overline{x}$ – mean; n – data number

(%i1) load (descriptive)$

(%i2) k: [0.021705365,0.040225261],
[2.175768116,1.26307321],
[3.448723255,3.032984714],
[3.898781151,5.913113435],
[4.320785997,9.967819791],
[4.924920596,6.090104586],
[5.601855942,2.968624296],
[6.540301992,1.777956557],
[7.941326922,0.796460177],
[9.97836439,0.056315366])$

(%i3) kurtosis (k), numer;

(%o3) –2.0

---

lagrange (list, option)

returns the polynomial interpolation by the Lagrange method. To use this function first call, load(interpol)$.

The following example is the interpolation at irregular (unequally spaced) intervals for: y=f(x)=sin(x).

x	21	25	26	31	22
y=sin(x)	0.3584	0.4226	0.4384	0.5150	?

For further details on Lagrange interpolation refer: Basics of Mathematical modeling for Science & Engineering Numerical Data Analysis, Dr. M Kanagasabapathy, Tech-Center, Amazon Co., (USA) 2013 I Edition, ISBN: 978-1492983125.

(%i1) load(interpol)$

(%i2) a:[[21, 0.3584], [25, 0.4226], [26, 0.4384], [31, 0.5150]]$

(%i3) lagrange(a);

(%o3) 0.001716666666666667*(x–26)*(x–25)*(x–21)–
0.017536*(x–31)*(x-25)*(x–1)+
0.01760833333333333*(x–31)*(x–26)*(x–21)–
0.001792*(x–31)*(x–26)*(x–25)

(%i4) f(x):=''%;

(%o4) f(x):=0.001716666666666667*(x–26)*(x–25)*(x–21)–
0.017536*(x–31)*(x–25)*(x-1)+0.01760833333333333
(x–31)(x–26)*(x–21)–0.001792*(x-31)*(x-26)*(x–25)

(%i5) f(22); /* interpolation for sin(22) */

(%o5) 0.374564

---

lambda ([i], function(i))

returns a lambda expression for the given defined function.

(%i1) f: lambda ([x], x^3+2);

(f) lambda ([x], x^3 +2)

(%i2) f(2*i);

(%o2) $8i^3+2$

(%i3) f(3);

(%o2) 29

---

laplace (expression, t, s)

returns the Laplace transform of expression with respect to the variable ‘t’ and transform parameter ‘s’.

(%i1) laplace (cos(a*t),t,s);

(%o1) $\frac{s}{s^2+a^2}$

(%i2) laplace (sin(a*t),t,s);

(%o2) $\frac{a}{s^2+a^2}$

(%i3) laplace (exp(a*t),t,s);

(%o3) $\frac{1}{s-a}$

(%i4) laplace (diff(f(t),t),t,s);

(%o4) s laplace(f(t),t,s)–f(0)

---

last (expression) **lastn (expression, count)**

returns the last expression or the count from the list.

(%i1) u:[–i/n, 2*j^k, 3*s^–g, exp(–u/n), z^k*y]$

(u) $[-\frac{i}{n}, 2j^k, \frac{3}{s^g}, \%e^{-\frac{u}{n}}, yz^k]$

(%i2) m:last (u)*i;

(m) iyz^k

---

lcm (expression)

returns the least common multiple for expression(s) or numbers. To use this function first call, **load ("functs")$**.

(%i1) load ("functs")$

(%i2) lcm(–n/r,–r/n,–3*g/(4*u));

(%o2) –3gnr

(%i3) lcm(3,21);

(%o3) 21

(%i6) a:x^3+4*x^2+4*x;

b:2*x^3+5*x^2+2*x;

c:x^4–x^2+2*x^3–2*x;

(a) $x^3 + 4x^2 + 4x$

(b) $2x^3 + 5x^2 + 2x$

(c) $x^4 + 2x^3 - x^2 - 2x$

(%i7) lcm(a, b, c);

(%o8) $(x-1)x(x+1)(x+2)^2(2x+1)$

---

ldefint (expression, x, lower_limit, upper_limit)

returns the definite integral of expression with respect to ‘x’ between the upper limit ‘b’ and the lower limit ‘a’.

(%i1) ldefint (2*x^4, x, a, b);

(%o1) $\frac{2b^5}{5} - \frac{2a^5}{5}$

(%i2) ldefint ((2*x)^4, x, a, b);

(%o3) $\frac{16b^5}{5} - \frac{16a^5}{5}$

---

ldisp (expression) **ldisplay (expression)**

returns expression into intermediate expressions and returns the list of labels.

(%i1) a:((n–r)^3);

(a) $(n-r)^3$

(%i2) b:expand(a);

(b) $-r^3 + 3nr^2 - 3n^2r + n^3$

(%i3) ldisp (a,b);

(%t3) $(n-r)^3$

(%t4) $-r^3 + 3nr^2 - 3n^2r + n^3$

(%o4) [%t3,%t4]

---

legendre_p (n, x);

returns Legendre polynomial of first kind of degree 'n'. To use this function first call, **load ("orthopoly")$**.

(%i1) load ("orthopoly")$

(%i2) legendre_p (2, x); /* for II order polynomial */

(%o2) $-3(1-x) + \frac{3(1-x)^2}{2} + 1$

(%i3) legendre_p (4, x); /* for IV order polynomial */

(%o4) $-10(1-x) + \frac{35(1-x)^4}{8} - \frac{35(1-x)^3}{2} + \frac{45(1-x)^2}{2} + 1$

---

legendre_q (n, x);

returns Legendre polynomial of second kind of degree 'n'. To use this function first call, **load ("orthopoly")$**.

(%i1) load ("orthopoly")$

(%i2) legendre_q (2, x); /* for II order polynomial */

(%o2)/R/ $\frac{3\log\left(-\frac{x+1}{x-1}\right)x^2 - 6x - \log\left(-\frac{x+1}{x-1}\right)}{4}$

---

length (expression)

returns the number of parts in the expression.

(%i1) u:[−i/n, 2*j^k, 3*s^−g, exp(−u/n), z^k*y]$

(u) $[-\frac{i}{n}, 2j^k, \frac{3}{s^g}, \%e^{-\frac{u}{n}}, yz^k]$

(%i2) v:length(u)+i;

(v) i+5

---

let (x, r) **letrat** **letsimp (expression(x))**

defines a substitution rule for the function 'letsimp' such that 'x' is 'r'. and 'x' is a product of powers. And the function 'matchdeclare' for 'x' is set to true. Expression at letsimp can be executed by setting the function 'letrat' to true.

(%i1) matchdeclare (a, true)$

(%i2) let (a/(a^2), b);

(%o2) $\frac{1}{a} \rightarrow b$ /* $\frac{1}{a} = b$ */

(%i3) letrat: true$

(%i4) letsimp (a/(a^4)); /* $\frac{a}{a^4} = \frac{1}{a^3}$ */

(%o4) b^3

---

lfreeof ([list], expression)

returns false, if any call to freeof function does. Refer 'freeof' function.

(%i1) lfreeof ([i, o], x^(a));

(%o2) true

(%i2) lfreeof ([i, o], i^b);

(%o2) false

(%i3) lfreeof ([i, o], b^i);

(%o3) false

---

limit (expression, variable, value)

returns limit of expression for the variable approaches the value.

(%i1) limit(log(x), x, 1);

(%o1) 0

(%i2) limit(sin(x), x, 0);

(%o2) 0

(%i3) limit(log(x), x, %e); /* %e = 2.718281828459045 */

(%o3) 1

---

linear (expression, variable)

returns a list of three equations for the variable, if expression of the linear form a*x + b where a ≠ 0 and a and b are independent of x. To use this function first call, **load (antid)**.

(%i1) load (antid");

(%o1) "C:\maxima-5.38.1\share\maxima\5.38.1_5_gdf93b7 b_dirty\share\integration\antid.mac"

(%i2) linear (((1– a)*(1+b))*(x)*c, x);

(%o2) [bargumentb=0,aargumenta=((1–a)*b–a+1)*c, xargumentx=x]

---

linear_regression (x)

estimates the linear regression for ‘x’ with confident level 95%. To use this function first call, **load(“stats”)$.**

(%i1) load(“stats”);

(%o1) "C:\maxima-5.38.1\share\maxima\5.38.1_5_gdf93b7 b_dirty\share\stats\stats.mac"

(%i2) a:matrix([1.00,1.00],[2.00,2.00],[3.00,1.30],[4.00,3.75], [5.00,2.25]);

(a) $\begin{bmatrix} 1.0 & 1.0 \\ 2.0 & 2.0 \\ 3.0 & 1.3 \\ 4.0 & 3.75 \\ 5.0 & 2.25 \end{bmatrix}$

(%i3) linear_regression(a);

(%o3)

LINEAR REGRESSION MODEL

$b_{estimation}$ = [0.7850000000000019,0.4249999999999998]

$b_{statistics}$ = [0.7760210016181804,1.393442622950819]

b_p_values = [0.4942986846956039,0.2577773028538888]

$b_{distribution}$ = [$student_t$, 3]

$v_{estimation}$ = 0.9302499999999999

v_conf_int = [0.2985269055625475,12.93239576911373]

$v_{distribution}$ = [chi2,3]

adc = 0.1905590602566893

---

linearinterpol ([matrix])

computes linear polynomial interpolation. To use this function first call, **load(interpol)$**.

(%i1) load(interpol)$

(%i2) x:matrix([2.1,6.5], [3.1,8.5], [3.4,9.1], [5.2,12.7], [6.5,15.3], [8.2,18.7], [8.5,19.3]);

(x) $\begin{bmatrix} 2.1 & 6.5 \\ 3.1 & 8.5 \\ 3.4 & 9.1 \\ 5.2 & 12.7 \\ 6.5 & 15.3 \\ 8.2 & 18.7 \\ 8.5 & 19.3 \end{bmatrix}$

(%i3) linearinterpol(x);

(%o3) (2.0*x+2.3)*charfun2(x,–inf,3.1)+(2.0*x+2.300000000000001)*charfun2(x,8.2,inf)+(2.0*x+2.300000000000001)*charfun2(x,6.5,8.2)+(2.000000000000001*x+2.299999999999992)*charfun2(x,5.2,6.5)+

(2.0*x+2.300000000000001)*charfun2(x,3.4,5.2)+(2.0*x+2.3)*charfun2(x,3.1,3.4)

(%i4) f(x):=''%;

(%o4) f(x):=(2.0*x+2.3)*charfun2(x,–inf,3.1)+(2.0*x+2.300000000000001)*charfun2(x,8.2,inf)+(2.0*x+2.300000000000001)*charfun2(x,6.5,8.2)+(2.000000000000001*x+2.299999999999992)*charfun2(x,5.2,6.5)+(2.0*x+2.300000000000001)*charfun2(x,3.4,5.2)+(2.0*x+2.3)*charfun2(x,3.1,3.4)

(%i5) f(2.8); /* interpolate at x = 2.8 */

(%o5) 7.9 f(x) = 2.0x+2.3 = (2.0×2.8)+2.3

---

linsolve ([equations], [variables])

solves simultaneous linear equations for the list of variables.

(%i1) linsolve([x+y=–1, 3*x–y=–11], [x,y]);

(%o1) [x=–3,y=2]

---

linechar

it is the prefix for the labels of intermediate expressions. Default value is '%t'. It can be changed with this function.

(%i1) a:((n–r)^3);

(a) $(n-r)^3$

(%i2) b:expand(a);

(b) $-r^3 + 3nr^2 - 3n^2r + n^3$

(%i3) ldisp (a,b); /* Refer **ldisp** function */

(%t3) $(n-r)^3$ /* default linecahr: 't' */

(%t4) $-r^3 + 3nr^2 - 3n^2r + n^3$

(%o4) [%t3,%t4]

call, **load ("lrats")$.**

(%i1) load ("lrats");

(%o1) "C:\maxima-5.38.1\share\maxima\5.38.1_5_gdf93b7 b_dirty\share\simplification\lrats.mac"

(%i2) lratsubst ([a^n = b, g/n = b, 1/r=i], a^n + g/n–1/r);

(%o2) 2b–i

---

lreduce (f, [List])

extends the binary function 'f' to the list.

(%i1) lreduce (f, [n,j]);

(%o1) f(n,j)

---

lsquares_estimates ([List], [Variables], Equation, [Coefficients])

returns the least square best fit coefficients from the given list or matrix for the proposed polynomial equation containing variables, which are given as list. To use this function first call, **load(lsquares)$**.

(%i1) load(lsquares)$

(%i2) k:matrix([1.0,15.6],[2.0,17.5],[3.0,36.6],[4.0,43.8], [5.0,58.2], [6.0,61.6],[7.0,65.2],[8.0,72.6],[9.0,98.9]);

(k) $\begin{bmatrix} 1.0 & 15.6 \\ 2.0 & 17.5 \\ 3.0 & 36.6 \\ 4.0 & 43.8 \\ 5.0 & 58.2 \\ 6.0 & 61.6 \\ 7.0 & 65.2 \\ 8.0 & 72.6 \\ 9.0 & 98.9 \end{bmatrix}$

/* List of independent and its dependent values are given as matrix*/

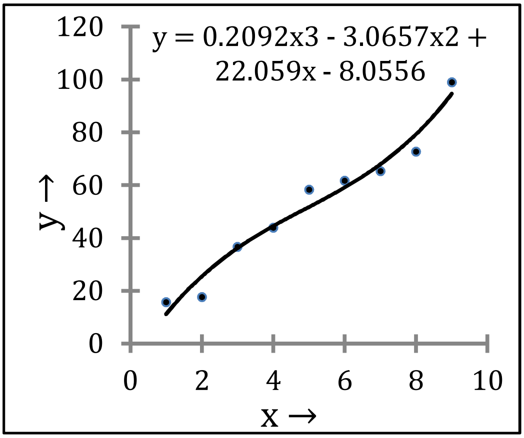
(%i3) lsquares_estimates(k, [x, y], y = a*x^3 + b*x^2+c*x+d, [a,b,c,d]); /* 3rd order polynomial curve fit */

(%o3) $[[a=\frac{497}{2376},b=-\frac{607}{198},c=\frac{52411}{2376},d=-\frac{145}{18}]]$

(%i4) %,numer;

(%o4) [[a=0.2091750841750842,b=–3.065656565656566,
c=22.05850168350169,d=–8.055555555555555]]

Based on the above data, the 3rd polynomial curve fit (trendline option) using spread sheet is sketched below:



lsquares_mse ([list], [variables], equation)

returns the mean square error, for the equation with the variables for the list. To use this function first call, **load(lsquares)$**.

(%i1) load(lsquares)$

(%i2) k:matrix([1.0,15.6],[2.0,17.5],[3.0,36.6],[4.0,43.8],
[5.0,58.2], [6.0,61.6],[7.0,65.2],[8.0,72.6],[9.0,98.9]);

(k) $\begin{bmatrix} 1.0 & 15.6 \\ 2.0 & 17.5 \\ 3.0 & 36.6 \\ 4.0 & 43.8 \\ 5.0 & 58.2 \\ 6.0 & 61.6 \\ 7.0 & 65.2 \\ 8.0 & 72.6 \\ 9.0 & 98.9 \end{bmatrix}$

(%i3) lsquares_mse (k, [x, y], x+y = a*x^3 + b*x^2+c*x+d);

(%o3) $\sum_{i=1}^{9} \frac{\left(k_{i,2}-ak_{3}^{i,1}-bk_{2}^{i,1}-ck_{i,1}+k_{i,1}-d\right)^2}{9}$

---

lsquares_estimates_exact (mean square error, coefficient)

returns the coefficients for the given data from mean square error. To use this function first call, **load(lsquares)$**.

Refer: lsquares_mse (List, [Variables], Equation)

(%i1) load(lsquares);

(%o1) "C:\maxima-5.38.1\share\maxima\5.38.1_5_gdf93b7 b_dirty\share\lsquares\lsquares.mac"

(%i2) k:matrix([1.0,15.6],[2.0,17.5],[3.0,36.6],[4.0,43.8], [5.0,58.2], [6.0,61.6],[7.0,65.2],[8.0,72.6],[9.0,98.9])$

/ * Refer: '**lsquares_mse**' for this data matrix output */

(%i3) lsquares_mse (k, [x, y], x+y = a*x^3 + b*x^2+c*x+d);

(%i4) lsquares_estimates_exact (k, [a, b, c, d]);

(%o3) $\sum_{i=1}^{9} \frac{\left(k_{i,2}-ak_{3}^{i,1}-bk_{2}^{i,1}-ck_{i,1}+k_{i,1}-d\right)^2}{9}$

---

lsum (expression, variable, List)

returns the sum of expression for the list from the variable.

(%i1) lsum (x*j, j, [−n, k, m]);

(%o1) −nx+mx+kx

(%i2) lsum (2*j, j, [8,7]);

(%o2) 30 /* (2×8) + (2×7) */

---

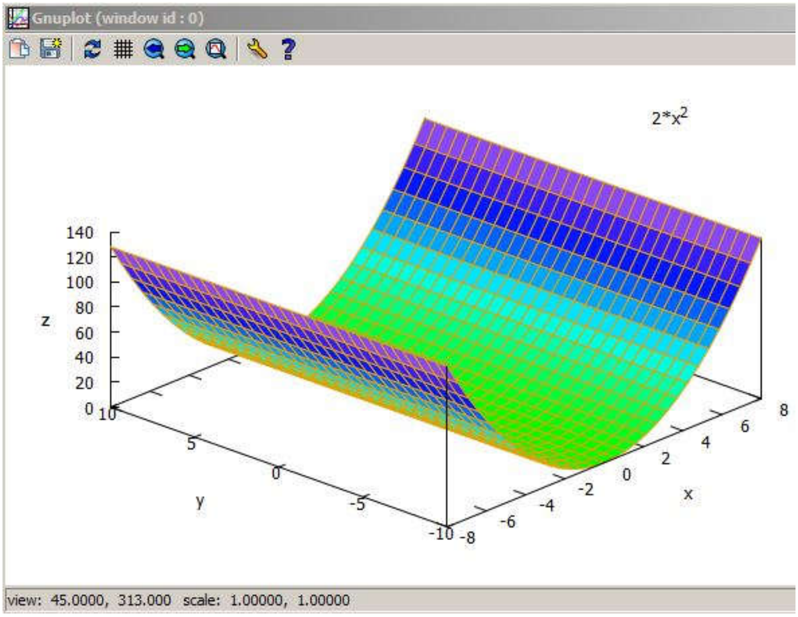
ltreillis (integer, length)

returns the list of partitions for integer, with the specified length.

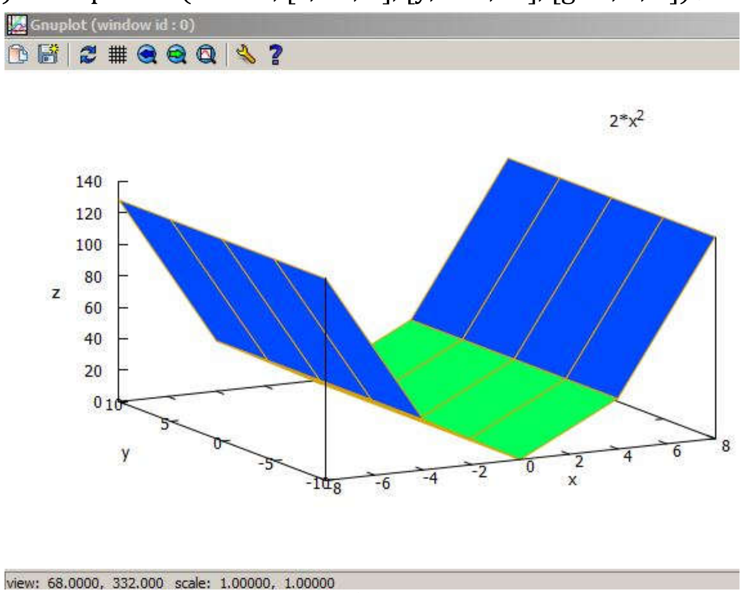
plot3d Few fundamental functions and plotting options related to 3d plot, through Gnuplot are outlined here.

(%i1) plot3d (2*x^2 , [x, –8, 8], [y,–10, 10])$

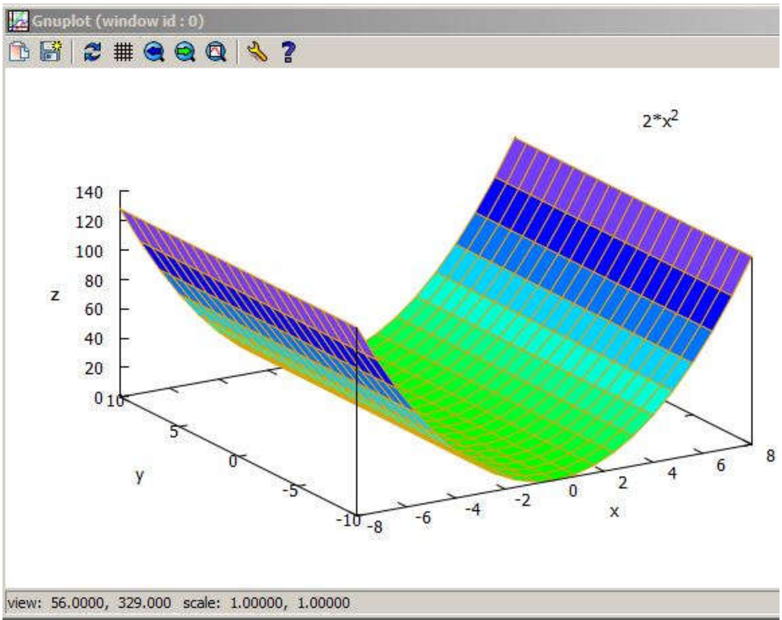
/* all 3D plots can be viewed by rotating at various angles */



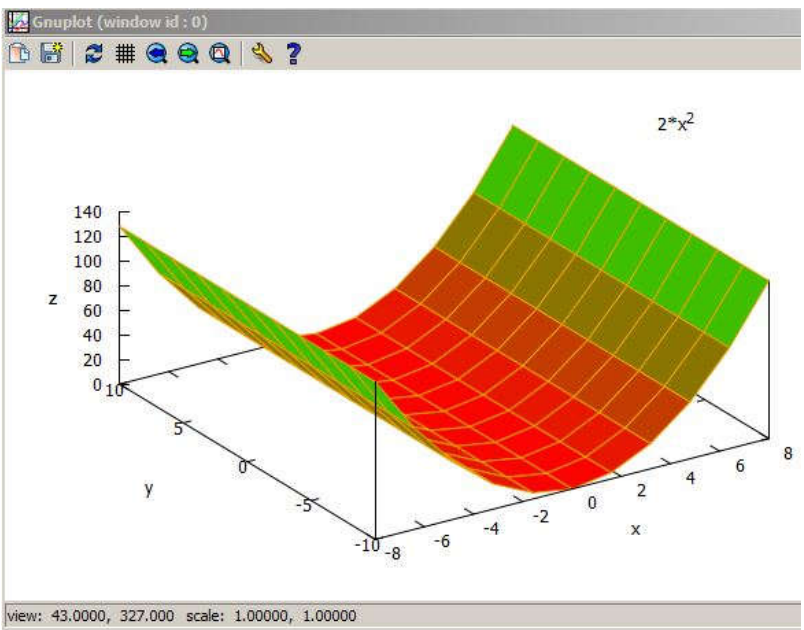
(%i1) plot3d (2*x^2, [x, –8, 8], [y, –10,10], [grid, 4, 4])$



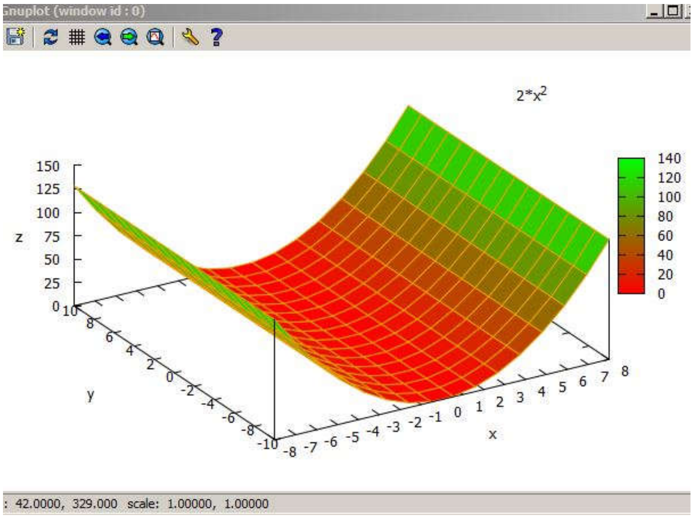
(%i1) plot3d (2*x^2, [x, −8,8], [y, −10,10], [grid, 20, 20])$



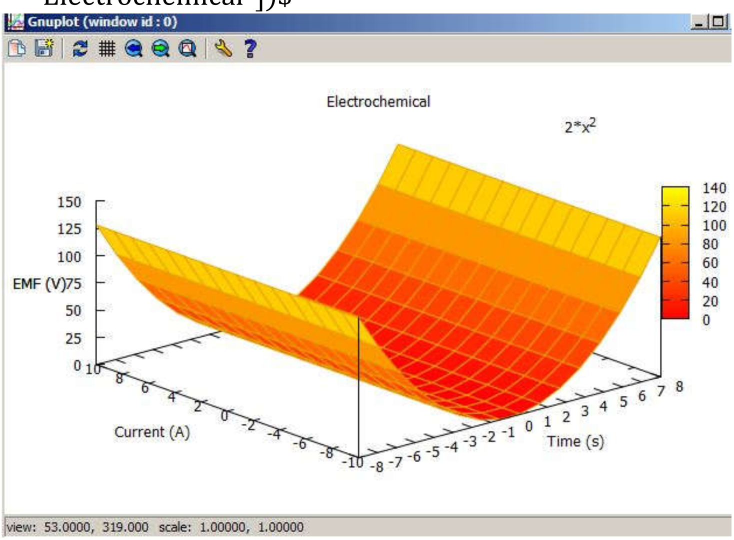
(%i1) plot3d (2*x^2, [x, −8, 8], [y, −10, 10], [grid, 10, 10], [palette, [gradient, red, green]])$ /* options */



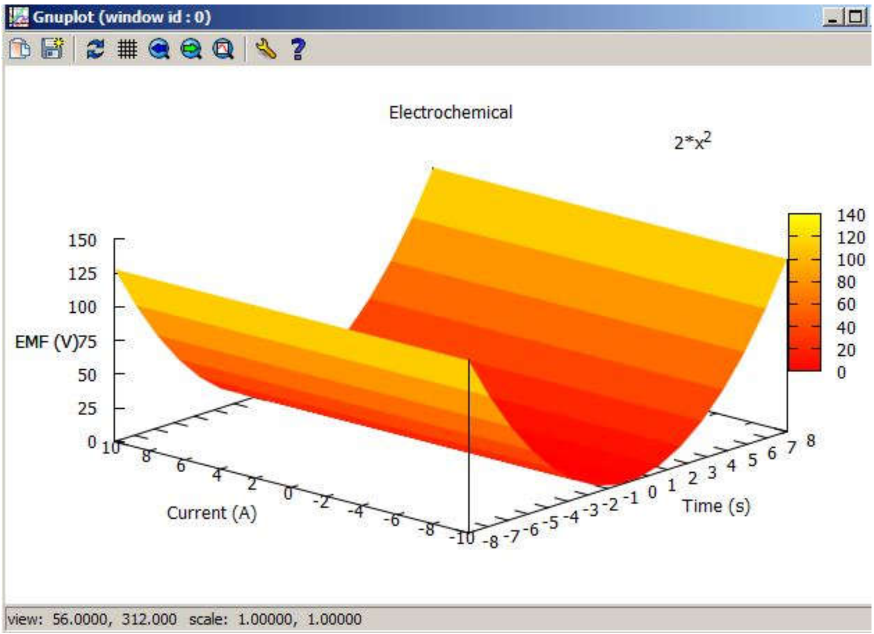
(%i1) plot3d (2*x^2 , [x, −8, 8], [y, −10, 10], [grid, 15, 15], [palette, [gradient, red, green]], color_bar, [xtics, 1], [ytics, 2], [ztics, 25],[color_bar_tics, 20])$



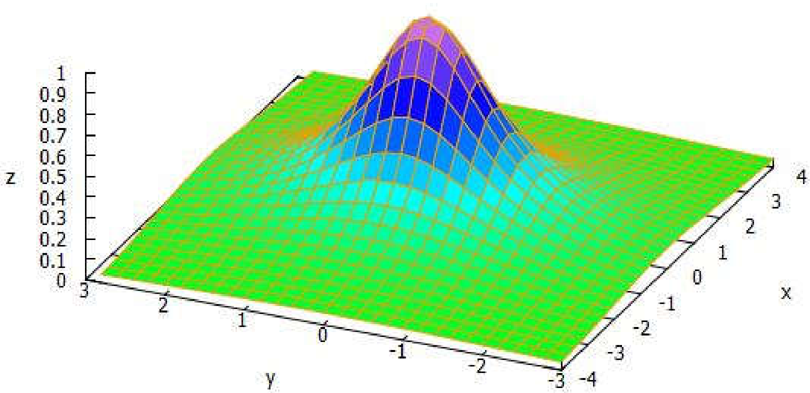
(%i1) plot3d (2*x^2 , [x, −8, 8], [y, −10, 10], [grid, 15, 15], [palette, [gradient, red, yellow]],color_bar, [xtics, 1], [ytics, 2], [ztics, 25],[color_bar_tics, 20], [xlabel, "Time (s)"], [ylabel, "Current (A)"],[zlabel, "EMF (V)"], [title, "Electrochemical"])$



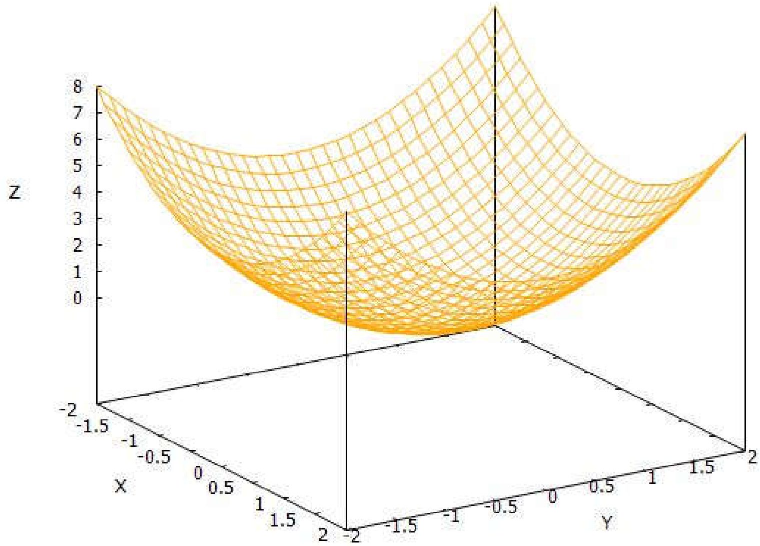
(%i1) plot3d (2*x^2, [x, −8, 8], [y, −10, 10], [grid, 15, 15], [mesh_lines_color, false], [palette, [gradient, red, yellow]], color_bar, [xtics, 1], [ytics, 2], [ztics, 25],[color_bar_tics, 20], [xlabel, "Time (s)"], [ylabel, "Current (A)"], [zlabel, "EMF (V)"], [title, "Electrochemical"])$



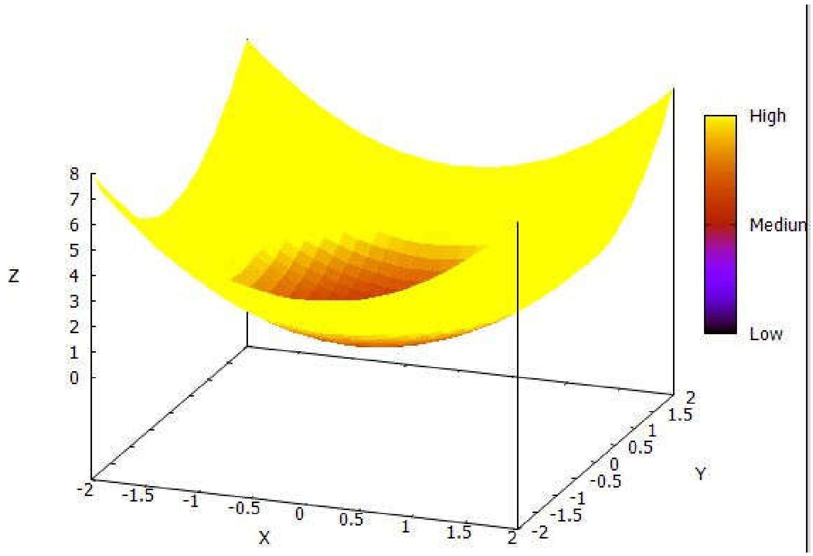
(%i1) plot3d(1/(1+x^2+y^2),[x,−4,4],[y,−3,3])$



(%i1) draw3d (color = orange, enhanced3d = false, cbtics = {["High",2],["Medium",0],["Low",–2]},cbrange=[–2, 2],explicit(x^2+y^2, x,–2,2,y,–2,2)) $



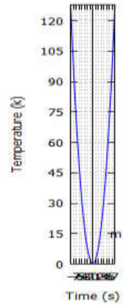
(%i1) draw3d (color = yellow, enhanced3d = true, cbtics = {["High",2],["Medium",0],["Low",–2]},cbrange=[–2, 2],explicit(x^2+y^2, x,–2,2,y,–2,2)) $



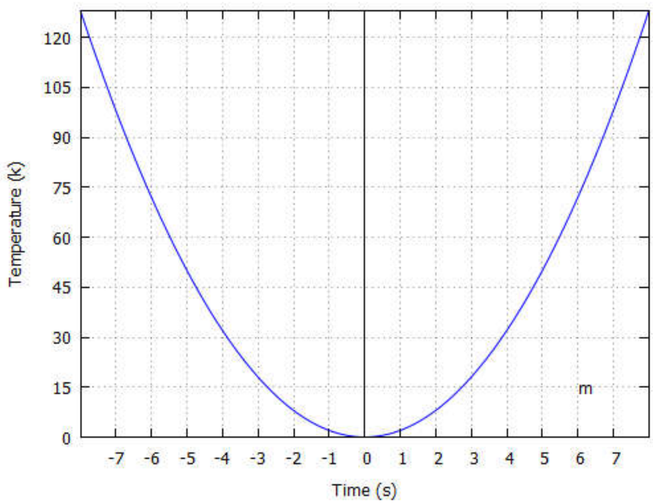
same_xy

if it is set to true, the scales used in the 'x' and 'y' axes will be the same, in 2d or 3d plots.

(%i1) plot2d (2*x^2, [x, −8, 8], [xtics, −7,1,7], [ytics, 0,15, 120],[axes, solid],grid2d,[**same_xy, true**],[label, ["m", 6, 15]],[xlabel, "Time (s)"], [ylabel, "Temperature (k)"])$



(%i1) plot2d (2*x^2, [x, −8, 8], [xtics, −7,1,7], [ytics, 0,15, 120],[axes, solid],grid2d,[**same_xy, false**],[label, ["m", 6, 15]],[xlabel, "Time (s)"], [ylabel, "Temperature (k)"])$



---

same_xyz

same as 'same_xy' function but for 3d plots.