Using MAXIMA in the Mathematics Classroom

Eugenio M. Fedriani * Rafael Moyano †

23th April, 2010

Abstract

Coming from the MACSYMA system and adapted to the Common Lisp standard, the symbolic calculus program called Maxima can be regarded as a tool for a frequent use in the maths classroom. Several circumstances may lead us to this consideration.

Firstly, this is a free software included in the Guadalinex distribution, supported by the Junta de Andalucía, which makes it easily available at almost every school in Andalusia. Secondly, Maxima not only is able to manipulate and simplify algebraic expressions but also is a programming language which allows us to create our own specific applications. Besides, it has a lot of packages as a complement, making it suitable for working in a lot of mathematical branches. And finally, this tool has been developed under several operative systems, in particular, it can be implemented under Windows, Linux and Macs, which can contribute to spread its use.

The main aim of this work is to show some of the possibilities of Maxima and its graphical interface as a tool for teaching Mathematics at the University as well as at a secondary school. Mathematics in Business degrees and A-levels can be provided with a resource that will make easier the learning of this subject to students.

As a conclusion, we also present a report of the main strengths and weaknesses of this software when used into the Mathematics classroom.

1 INTRODUCTION

The symbolic calculus program *Maxima* comes from the *Macsyma* system (MAC's SYmbolic MAnipulation), which was developed by the MIT (Massachusetts Institute of Technology) from 1968 to 1982 as a part of the MAC project (Machine Aided Cognition). The MIT provided a copy of the source code to the DOE (Department of Energy) in 1982, within a version known as DOE-Macsyma. Later on, the DOE granted the exploitation rights to the Symbolics company,

*Department of Economics, Quantitative Methods and Economic History. University of Pablo de Olavide. Seville. e-mail: efedmar@upo.es

†Department of Economics, Quantitative Methods and Economic History. University of Pablo de Olavide. Seville. e-mail: rmoyfra@upo.es

which still developed the project for some years more. In 1992 the program was acquired by a company called Macsyma Inc. However, its commercial exploitation was becoming less interesting than before due to the appearance of some other programs, like Maple or Mathematica, with a more friendly working environment, although deriving from Macsyma.

Simultaneously, Professor William F. Schelter from Texas Univesity kept one of the copies of the program provided by the MIT from 1982 to his death, in 2001. This version was adapted to the common Lisp standard, and was called *Maxima* to distinguish it from the commercial version. In 1998, Shelter was allowed to spread out the source code of the DOE-Macsyma under GNU-GLP license and, in 2000, he initializes the Maxima project in SourceForge, aiming to keep and develop DOE-Macsyma under the name of Maxima.

Since Maxima is distributed under the GNU-GLP license, both source code and manuals are for free access through the website of the project, (*maxima.sourceforge.net*).

2 INSTALLING MAXIMA

Versions of Maxima are available under Linux and under Windows. In this paper only the Windows environment is considered. However, detailed information about the installation of Maxima under Linux can be found in the website of the project. This includes directives on the basic package as well as the additional modules to use the program in a basic context.

The Windows version is also available in the website of the project, where an executable file can be freely downloaded. The program is automatically installed once this file is executed. In this paper, version 5.17.1 has been used, provided by the file *maxima-5.17.1.exe*. However, further versions are already available. The executable file not only installs the kernel but also a collection of packages that simplify the working environment. The installation outcome includes a text console (MSDOS mode) and two graphical environments: *xMaxima* (with permanent help on screen, but only useful in command mode) and *wxMaxima*. This last is the most advisable one, since it allows to work in screen mode, incorporates a text editor and includes a wide collection of displayable menus as well as shortcuts and buttons to do a lot of mathematical computations without using commands. Besides, two modules of graphical representation are included: *gnuplot* and *openmath*.

Unlike some other free distribution programs, all the necessary modules and packages are automatically installed from the executable file, that is, you do not need to download and install one by one.

3 PRELIMINARIES

In this paper, we propose several tasks. All of them are supposed to be performed by using *wxMaxima*. Besides, pop-up menus and shortcuts are not going to be detailed in the following pages, since they can be easily deduced when used.

Let us recall that Maxima syntax uses the standard Lisp, from which it was developed. This syntax may be a bit odd to users which are not familiar with the Lisp language.

3.1 Basic Rules

1. The communication with the Kernel is in question-answer mode. The question can be simple (direct order) or multiple (program).

2. The desktop is divided into *cells*, and any written character is always in a cell. Cells can be *text cells* or *input cells*.

3. Input cells can contain a command, an operation or a program, as well as several of them. They can be written next or beneath each other, but every command or direct order must finish with <;> or < $ > (as a courtesy of Lisp). When finished with <;>, they generate an output showed on screen, whereas those with a final < $ > are internally executed, without generating any output on screen.

4. Inputs generate a label like this: (%*in*), and the corresponding label to outputs is: (%*on*), where n stands for the n-th execution. Both statements can be called by using their label.

5. A shortcut to execute an input cell is <SHIFT> + <INTRO>. Note that <INTRO> is only useful to add a new line within a cell.

6. An input with a single order can be directly executed. By the way, wxMaxima automatically includes the final <;>.

7. The result of any computation will be displayed in a simplified symbolic form, if possible. Any approximation can be requested, similarly to any other form of simplification. If the exact solution is not possible (for instance, when solving certain algebraic equations), Maxima will automatically use different numerical algorithms to find approximated solutions.

8. When a command is chosen from a menu or a button is pressed, a cell is automatically generated. Then the command that develops the desired action is written in that cell and in a suitable way.

9. A cell can be rewritten and executed as many times as desired. Cells can be inserted anywhere throughout the desktop. However, note that wxMaxima has a word editor, but not a word processor, so formatting text is not possible.

10. Work sessions can be saved (*File/Save*) or exported to HTML or TEX formats (*File/Export*). A saved session only contains non-executed inputs. Outputs need to be generated again when the saved session is restored.

11. In order to delete a cell, it must be selected (through its right bracket) before pressing <SUPR>. An output associated to a given input cannot be deleted independently, but it can be hidden by clicking the upper triangle in the cell's bracket.

3.2 Help

For a general help, using the help menu is strongly recommendable. However, some help about specific commands can be obtained by using *? command* or *describe(command)*

3.3 Entering Expressions

Elemental operators: $< + >$, $< - >$, $< * >$, $< / >$,and $< \hat{} >$ or $< ** >$

Variables assignation: <:>

Variable names: any alphanumerical combination starting with a letter is possible, except some reserved words (commands, built-in functions, system variables, etc).

Deleting variables from memory: *Maxima/Delete variable*. The default option, *all*, cleans the entire memory (not the screen), even the cells' labels.

Some constants: *%e*, *%pi*, *%i* (the imaginary unit), *inf* (real positive infinity), *minf* (real negative infinity).

Some elemental functions: *sqrt(x)*, *sin(x)*, *cos(x)*, *tan(x)*, *log(x)* (base *%e* logarithm; the remaining logarithms can be defined 'handly', by using the *change of base* formula), *exp(x)* (that is, *%e^x*), *abs(x)*, !, *ceiling(x)* (integer part of x, by excess), *floor(x)* (integer part of x, by defect).

Comments in an *input* **cell:** expressions like */* remark */* are not evaluated when executing an input cell.

4 BASIC ARITHMETIC OPERATIONS

```
(%i1) ((3^7/(4+3/5))^(-1)+7/9)^3 /*exact display*/;
```

(%o1)
$$\frac{620214013952}{1307544150375}$$

```
(%i2) ((3^7/(4+3/5))^(-1)+7/9)^3,numer /*approximated display*/;
```

(%o2) 0.47433504541634

The desired number of figures to approximate the output can be chosen in *Numeric/Set precision....*

```
(%i3) fpprec:500$
(%i4) ((3^7/(4+3/5))^(-1)+7/9)^3,bfloat /*computing again*/;
```

(%o4) $4.74335045416343576596531106 63[444 digits]88078741933471593884572197 7b - 1$

All the computed figures can be displayed by changing the screen mode (*Maxima/Change 2d display/ascii*). The default mode can be restored through the same command (*Maxima/Change 2d display 2D/xml*).

```
(%i5) set\_display('ascii)$
(%i6) ((3^7/(4+3/5))^(-1)+7/9)^3,bfloat /*approximated display*/;
```

(%o6) 4.7433504541634357659653110663704994971764912783700770414606811628136950
9698381071004835361294061649478319245622130834275615807807823599740067018079255
5578651228838434089728891537889306203000112978116232353000403747456518844662955
4924505908196912727135185245809333117219407146628832624897742661816311519438085
7945413328506322330921008767393913017948405503760120020490287071619067201778884
4330087759083482642130052747512372885980836798326990488716865558024312537164334
42204264189988078741933471593884572197 7b − 1

(%i7) `set\_display('xml)$`

5 MATRIX OPERATIONS

We can use *Algebra/Enter matrix...* to enter a matrix. This is a different option from *Algebra/Generate matrix...*, which is more suitable to generate, for instance, its entries from a general term. Of course, matrices can also be named.

(%i8) `matrix([1,2,0],[2,1,0],[0,0,3] );`

(%o8)
$$\begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

(%i9) `a:%o8;`

(%o9)
$$\begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

Matrix operations: addition, $< + >$; opposite element with respect to the addition, $< - >$; scalar multiplication, $< * >$; matrix multiplication, $< . >$ (be careful: there exists another 'product' that can be applied to matrices, *a*b*, which will multiply matrices a and b entry by entry); inverse, *Algebra/Invert matrix*; determinant, *Algebra/Determinant*; transpose, *Algebra/Transpose matrix*; cofactor matrix, *Algebra/Adjoint matrix*.

Note that a command appears on screen in an input cell when executing any of the above orders. It is also possible to write directly this command instead of pulling down the correspondent menu. This is a faster way of working, although you need to memorize some words. Besides, this way of interacting increases the possibilities the program has, since not every command is defined in a pop-up menu. On the other hand, the direct use of commands may not be comfortable to students, so the decision on its use is up to the teacher.

Operations not available in menus: rank, *rank(matriz)*; the minor of the (i,j)-entry, *minor(matriz,i,j)*; the identity matrix of size n, *ident(n)*.

(%i10) `b:matrix([1,1,1],[1,1,1],[1,1,1]);`

(%o10)
$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

(%i11) `a+b; a-b; 3*a;`

(%o11)
$$\begin{pmatrix} 2 & 3 & 1 \\ 3 & 2 & 1 \\ 1 & 1 & 4 \end{pmatrix}$$

(%o12)
$$\begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

(%o13)
$$\begin{pmatrix} 3 & 6 & 0 \\ 6 & 3 & 0 \\ 0 & 0 & 9 \end{pmatrix}$$

(%i14) `a.b /*matrix multiplication*/; a*b /*entry by entry product*/;`

(%o14)
$$\begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{pmatrix}$$

(%o15)
$$\begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

(%i16) `transpose(a);adjoint(a); invert(a);determinant(a); rank(a);`

(%o16)
$$\begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

(%o17)
$$\begin{pmatrix} 3 & -6 & 0 \\ -6 & 3 & 0 \\ 0 & 0 & -3 \end{pmatrix}$$

(%o18)
$$\begin{pmatrix} -\frac{1}{3} & \frac{2}{3} & 0 \\ \frac{2}{3} & -\frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} \end{pmatrix}$$

(%o19) -9

(%o20) 3

6

(%i21) `c:ident(4);`

(%o21)
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Advanced operations: characteristic polynomial, *Algebra/Characteristic polynomial...*; eigenvalues, *Algebra/Eigenvalues* (it shows both eigenvalues and its multiplicities); eigenvectors, *Algebra/Eigenvectors* (the output is a list whose first element is another list containing all the eigenvalues with its multiplicities).

(%i22) `charpoly(a, x), expand;`

(%o22)
$$-x^3 + 5\,x^2 - 3\,x - 9$$

The *factorize* button can be used to factorize the former polynomial as well as the menu *Simplify/Factor expression*.

(%i23) `factor(-x^3+5*x^2-3*x-9);`

(%o23)
$$-(x-3)^2\,(x+1)$$

(%i24) `eigenvalues(a);`

(%o24)
$$[[3,-1],[2,1]]$$

(%i25) `eigenvectors(a);`

(%o25)
$$[[[3,-1],[2,1]],[1,1,0],[0,0,1],[1,-1,0]]$$

There exists a variable called *nondiagonalizable*. Its value is FALSE if the matrix whose eigenvalues were last calculated **is** diagonalizable, and TRUE in the opposite case. It may be useful to check whether our matrix is diagonalizable or not. Obviously, our example must produce a FALSE.

(%i26) `nondiagonalizable;`

(%o26)
$$false$$

When the 'nondiagonalizable' output is FALSE and, consequently, our matrix is diagonalizable, both the similarity transformation matrix P and the similar diagonal matrix D (such that $D = P^{-1} \cdot A \cdot P$) can be obtained by using the command *similaritytransform*. Actually, this command produces the same result as *eigenvectors*, but all the given eigenvectors are now unitary. Additionally, it assigns interesting values to two environmental variables: *rightmatrix* is redefined as the transformation matrix P, and *leftmatrix* takes the value of P^{-1}. Remember that loading the *eigen* package is needed to use this command.

(%i27) `load(eigen)$ similaritytransform(m);`

(%o27)
$$[[[3,-1],[2,1]],[\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0],[0,0,1],[\frac{1}{\sqrt{2}},\frac{-1}{\sqrt{2}},0]]$$

(%i27) `print(leftmatrix,".",m,".",rightmatrix,"=", leftmatrix.m.rightmatrix)$`

(%o27)
$$\begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

5.1 Classification of Quadratic Forms

A quadratic form can be classified by using its associated symmetric matrix, once it has been obtained. This matrix can be effectively classified by its eigenvalues or its leading principal minors, that can be obtained by using the command *submatrix(i1, ..., ik, matrix, j1, ..., jk)*. Its output is the resulting submatrix from removing rows $i1$, ..., ik, and columns $j1$, ..., jk.

(%i27) `q:matrix([1,2,3],[4,5,6],[7,8,9]);/*qsim is the associated symmetric matrix*/ qsim:1/2*(q+transpose(q));`

(%o28)
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{pmatrix} 1 & 3 & 5 \\ 3 & 5 & 7 \\ 5 & 7 & 9 \end{pmatrix}$$

(%i29) `/*it is indefinite */eigenvalues(qsim),numer;`
'rat' replaced 17.91647286716892 by 84727/4729 = 17.9164728272362
'rat' replaced 17.91647286716892 by 84727/4729 = 17.9164728272362
(%o29) [[-1.458236413618101,16.4582364136181,0],[1,1,1]]

(%i30) `/*computing rank to check if classifying by minors is possible*/ rank(qsim);`

(%o30) 2

(%i31) `/*The leading principal minors:*/ submatrix(2,3,qsim,2,3);submatrix(3,qsim,3); qsim; determinant(submatrix(2,3,qsim,2,3));determinant(submatrix(3,qsim,3)); determinant(qsim);`

(%o31)
$$(1)$$

(%o32)
$$\begin{pmatrix} 1 & 3 \\ 3 & 5 \end{pmatrix}$$

(%o33)
$$\begin{pmatrix} 1 & 3 & 5 \\ 3 & 5 & 7 \\ 5 & 7 & 9 \end{pmatrix}$$

(%o34) 1

(%o35) -4

(%o36) 0

Hence, classifying by leading minors is possible and the quadratic form remains indefinite, obviously.

(%i37)
```
/*polynomial form*/ [x,y,z].qsim.[x,y,z],expand;
```

(%o37) $$9\,z^2+14\,y\,z+10\,x\,z+5\,y^2+6\,x\,y+x^2$$

We have also another option to classify the quadratic form by leading principal minors. Although Maxima has no command to directly supply the leading principal minors of a given matrix, we can easily build a function to do this:

(%i31)
```
leadminors[m]:=reverse(block([l:[m],k:length(m)],for a:1 thru k-1 do
(m:submatrix(length(m),m,length(m)),l:append(l,[m])),l))$
leadpralminors[m]:=reverse(block([l:[determinant(m)],k:length(m)],
for a:1 thru k-1 do
(m:submatrix(length(m),m,length(m)),l:append(l,[determinant(m)])),l))$
```

In fact, we have defined two functions: *leadminors[m]* displays the submatrices whose determinants are defined as the leading principal minors, and *leadpralminors[m]* displays the minors themselves. When applied to our former matrix, the obtained result is the following:

(%i32)
```
leadminors[qsim];leadpralminors[qsim];
```

(%o32) $$[(1)\,,\begin{pmatrix}1&3\\3&5\end{pmatrix},\begin{pmatrix}1&3&5\\3&5&7\\5&7&9\end{pmatrix}]$$

(%o33) $[1,-4,0]$

We are using Maxima as a tool in the classroom, and students may find this task a little bit hard. However, we can easily save these functions in a file and load it as a package when needed. We will see how to do this later on.

6 VECTORS

Unlike some others symbolic calculus programs, matrices and vectors are two distinct types of objects for Maxima. A vector is a list of numerical values. And its components are introduced in brackets and separated with commas.

(%i38)
```
v1:[1,2,3,4];v2:[2,4,6,8];
```

(%o39) $[1,2,3,4][2,4,6,8]$

Operations: $<+>$, $<->$, $<*>$ (multiplication by scalars), $<.>$ (scalar product), *transpose(vector)* (transform a row vector into a column one).

(%i40) `v1.v2;-1/2*v1;transpose(v2);`

(%o40) 60

(%o41) $$[-\frac{1}{2}, -1, -\frac{3}{2}, -2]$$

(%o42) $$\begin{pmatrix} 2 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

7 SYSTEMS OF EQUATIONS

The most powerful command to solve equations as well as systems is *solve(eqns,vars)*, which is available from the button *Solve...* or the menu *Equations/Solve...*

To enter a system, the equations and their variables are separated by commas. This command is useful for linear and non-linear systems.

(%i43) `solve([x^3-2*x^2+1=0],[x]);`

(%o43) $$[x = -\frac{\sqrt{5}-1}{2}, x = \frac{\sqrt{5}+1}{2}, x = 1]$$

(%i44) `solve([x^2+y=2,x*y=1],[x,y]);`

(%o44) $$[[x = 1, y = 1], [x = -\frac{\sqrt{5}+1}{2}, y = -\frac{2}{\sqrt{5}+1}], [x = \frac{\sqrt{5}-1}{2}, y = \frac{2}{\sqrt{5}-1}]]$$

Equations/Solve linear system... and *Equations/Solve algebraic system...* are two more specific alternatives, but their specificity make difficult its use by students.

The *solve* command tries to find exact solutions but, if it fails, use numerical algorithms to obtain an approximation of every solution. If no algorithms run or the system has no solution, the output will be [].

When approximations are displayed, the exact solution might be found by using another way. In the following example we can find the exact solutions by the command *eliminate(eqns,vars)*.

(%i45) `solve([x^2-y^2=1,x*y-x=1],[x,y]);`

(%o45) $[[x = -1, y = 0], [x = 0.6062907292072\,i - 0.41964337760708, y = 0.22815549365396 - 1.115142508039937\,i], [x = -0.6062907292072\,i - 0.41964337760708, y = 1.115142508039937\,i + 0.22815549365396], [x = 1.839286758257819, y = 1.543689011792378]]$

(%i46) `eliminate([x^2-y^2=1,x*y-x=1], [x]);`

(%o46) $$[-y\,\left(y^3 - 2\,y^2 + 2\,y - 2\right)]$$

(%i47) `soly:solve(-y*(y^3-2*y^2+2*y-2)=0,y);`

$$(\%o47)\quad [y=-\frac{2\left(\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)}{9\left(3^{-\frac{3}{2}}\sqrt{11}+\frac{17}{27}\right)^{\frac{1}{3}}}+\left(3^{-\frac{3}{2}}\sqrt{11}+\frac{17}{27}\right)^{\frac{1}{3}}\left(-\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)+\frac{2}{3},$$
$$y=\left(3^{-\frac{3}{2}}\sqrt{11}+\frac{17}{27}\right)^{\frac{1}{3}}\left(\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)-\frac{2\left(-\frac{\sqrt{3}\,i}{2}-\frac{1}{2}\right)}{9\left(3^{-\frac{3}{2}}\sqrt{11}+\frac{17}{27}\right)^{\frac{1}{3}}}+\frac{2}{3},$$
$$y=\left(3^{-\frac{3}{2}}\sqrt{11}+\frac{17}{27}\right)^{\frac{1}{3}}-\frac{2}{9\left(3^{-\frac{3}{2}}\sqrt{11}+\frac{17}{27}\right)^{\frac{1}{3}}}+\frac{2}{3},y=0]$$

(%i48) `makelist(solve(ev(x*y-x=1,soly[i]),x),i,1,4);`

$$(\%o48)\quad [[x=-\frac{18\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{1}{3}}}{\left(9\sqrt{3}\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{2}{3}}+2\sqrt{3}\right)i+9\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{2}{3}}+6\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{1}{3}}-2}],$$
$$[x=\frac{18\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{1}{3}}}{\left(9\sqrt{3}\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{2}{3}}+2\sqrt{3}\right)i-9\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{2}{3}}-6\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{1}{3}}+2}],$$
$$[x=\frac{9\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{1}{3}}}{9\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{2}{3}}-3\left(\frac{\sqrt{11}}{3\sqrt{3}}+\frac{17}{27}\right)^{\frac{1}{3}}-2}],[\mathrm{x}=-1]]$$

Finally, let us see an example of solving an underdetermined system, finding its general solution as well as some particular ones.

(%i49) `solgen:solve([x+y-z=0,2*x-y=0], [x,y,z]);`

$$(\%o49)\quad [[x=\frac{\%r1}{3},y=\frac{2\,\%r1}{3},z=\%r1]]$$

(%i50) `solgen,%r1:3;solgen,%r1:-2;`

$$(\%o50)\quad [[x=1,y=2,z=3]]$$

$$(\%o51)\quad [[x=-\frac{2}{3},y=-\frac{4}{3},z=-2]]$$

7.1 Quadratic Forms Subject to Linear Constraints

We just need to solve the equation formed by the linear constraints, and substitute the general solution in the matrix form or in the polynomial expression of the quadratic form. As an example, let us consider the former quadratic form $q(x,y,z)=9z^2+14yz+10xz+5y^2+6xy+x^2$ subject to the linear constraints S defined as $S:\begin{cases} x+y-z &= 0\\ 2x-y &= 0\end{cases}$:

(%i52) `solve([x+y-z=0,2*x-y=0],[x,y]);`

$$(\%o52)\quad [[x=\frac{z}{3},y=\frac{2\,z}{3}]]$$

(%i53) `subst([x=z/3,y=(2*z)/3],9*z^2+14*y*z+10*x*z+5*y^2+6*x*y+x^2);`

(%o53)
$$\frac{76\,z^2}{3}$$

Hence, the former quadratic form subject to these linear constraints is positive definite.

8 GRAPHICAL REPRESENTATIONS

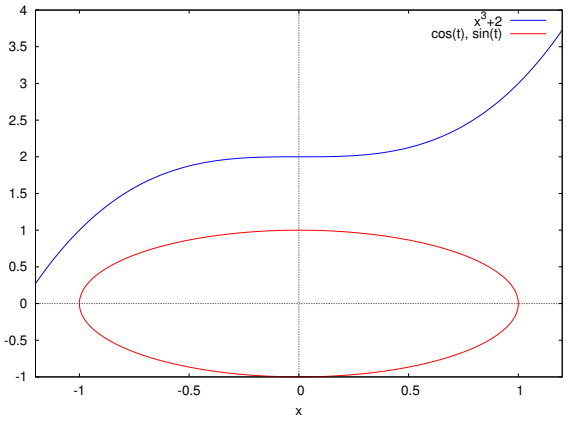
The shortest way to get a 2-dimensional graphical representation is using the button *Plot 2D...*

If a command is followed by an ellipsis, a dialog box is displayed when pressing its button. In the case we are dealing with, the most remarkable options are the following. *Expressions:* several graphics can be displayed at the same time; they must be separated by commas. *Special:* this button allows us to choose between parametric or discrete representations, that can be included all in the same graphics. *Format:* two different graphic modules are available: GNUPLOT and OPENMATH; both of them are freeware and they have been added to the Maxima distribution. GNUPLOT is more versatile, since OPENMATH does not admit certain representations. In addition, in this version of Maxima, graphics can be embedded in a line of our workfile (both modules display graphics on separated windows).

As an example, let us plot an explicit function and a parametric one in the same graphic.

(%i54) `wxplot2d([x^3+2, ['parametric, cos(t), sin(t), [t,-%pi,%pi], [nticks,300]]], [x,-1.2,1.2])$`

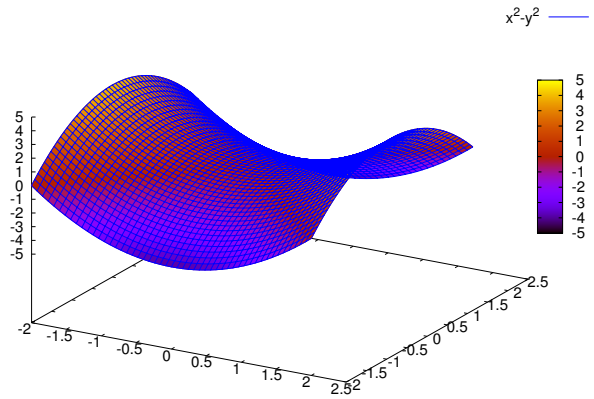
(%t54)



To plot 3-dimensional representations, we can use the *Plot 3D...* button.

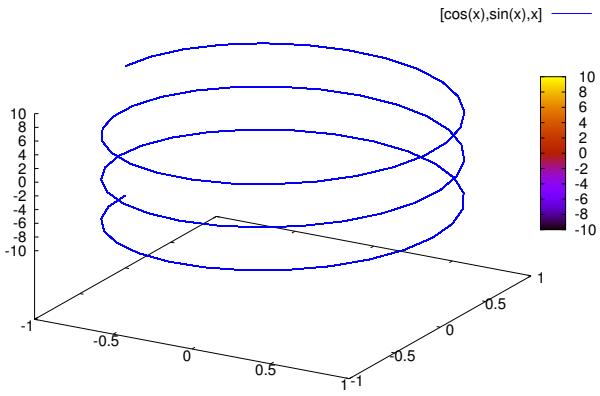
A dialog box similar to the former one is displayed, and the two above format options are available. However, unlike the 2D-representations, only one representation is admitted each time, and the *Special* button is not available. Therefore, parametric functions must be entered in the *Expression* line in a vectorial way. When GNUPLOT or OPENMATH options are chosen in separated windows, the obtained surface or curve can be rotated by dragging it, through the mouse.

(%i55) `wxplot3d(x^2-y^2,[x,-2,2],[y,-2,2], [plot\_format,gnuplot],[grid,50,50])$`



(%t55)

(%i56) `wxplot3d([cos(x),sin(x),x],[x,-3*%pi,3*%pi], [y,-3,3], [grid,80,80])$`



(%t56)

8.1 Contour Lines

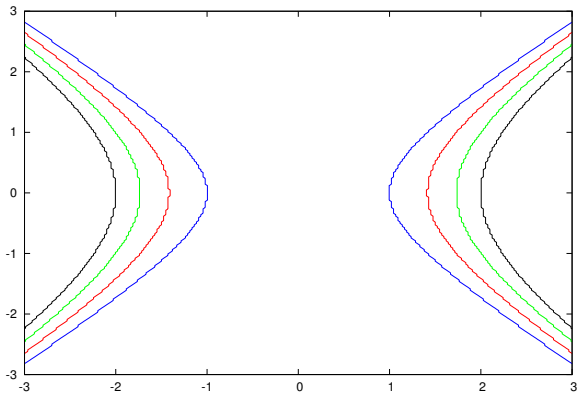
It is worth to mention graphical representations of level sets. When dealing with a function $f(x, y)$, we can 'manually' draw some different curves in the same graphic. If they are implicit, the package *draw* should be previously loaded, for instance, by using the menu *File/Load package...*, and the command *implicit(function,range,range)* can be applied within the *wxdraw2d(objects)* command.

(%i57) `a:makelist(implicit(x^2-y^2=k,x,-3,3,y,-3,3),k,1,4);`

(%o57) $[implicit\left(x^2-y^2=1,x,-3,3,y,-3,3\right),implicit\left(x^2-y^2=2,x,-3,3,y,-3,3\right),$
$implicit\left(x^2-y^2=3,x,-3,3,y,-3,3\right),implicit\left(x^2-y^2=4,x,-3,3,y,-3,3\right)]$

(%i58) `load("draw")$`

(%i59)
```
wxdraw2d(color=blue,a[1],color=red,a[2],color=green,a[3],color=black,a[4]);
```

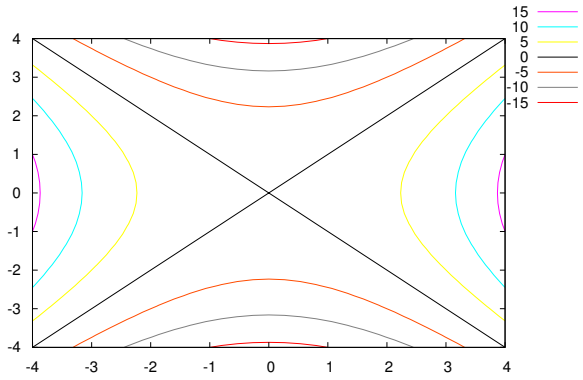


(%t59)

(%o59) $[gr2d\,(implicit, implicit, implicit, implicit)]$

This process can be automated by using the command *wxcontour_plot(function,[range],[range])*, which can work with implicit functions (besides, we have used the *set_plot_option* command to set the desired number of contour lines, but the description of its use is beyond the aim of this paper).

(%i60)
```
set\_plot\_option([gnuplot\_preamble,"set cntrparam levels 15"])$
wxcontour\_plot(x^2-y^2,[x,-4,4],[y,-4,4]);
```

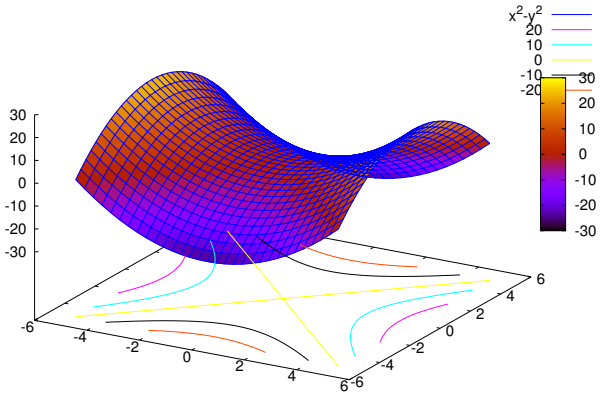


(%o61)

Note that any graphical command starting with *wx-* can also be written without this prefix, but generating the plotting on a separate window.

Finally, both the surface and its contour lines can be represented in the same graphic by adding the *[gnuplot_preamble, "set contour"]* option:

(%i62) `wxplot3d(x^2-y^2,[x,-5,5],[y,-5,5], [gnuplot\_preamble,"set contour"])$`



(%t62)

9 LIMITS AND SUMS

We do not need to distinguish between functions and sums to compute limits. On the other hand, most teaching needs can be satisfied by using the menu *Calculus/Find limit...* Recall that Maxima can distinguish between 'infinities with sign' (*inf, minf*) and 'without it' (*infinity*).

(%i63) `'limit(1/x,x,0);limit(1/x,x,0);`

(%o63) $$\lim_{x \to 0} \frac{1}{x}$$

(%o64) $$infinity$$

(%i65) `'limit(1/x, x, 0, minus);limit(1/x, x, 0, minus);`

(%o65) $$\lim_{x \to 0-} \frac{1}{x}$$

(%o66) $$-\infty$$

Notice that we have used the symbol <'> before some operators. Any operator preceded by <'> is not evaluated. This 'trick' can be used to show the desired limit written in a mathematical format.

We can get it even prettier by using the command *print*, but the only alternative is to write the following line by our own:

(%i67) `print('limit((n^3-7*n^2)/(n^2-3*n), n, inf),"=", limit((n^3-7*n^2)/(n^2-3*n), n, inf))$`

$$\lim_{n \to \infty} \frac{n^3 - 7\,n^2}{n^2 - 3\,n} = \infty$$

The menu *Calculus/Calculate sum...* also satisfies our teaching needs widely.

(%i68) `/*Direct computing*/ sum(7*n^2-3*n, n, 5, 15), simpsum;`

(%o68) 8140

(%i69) `'sum(1/2^k, k, 1, inf);sum(1/2^k, k, 1, inf), simpsum;`

(%o69) $$\sum_{k=1}^{\infty} \frac{1}{2^k}$$

(%o70) 1

To finish this section, let us remark that Maxima contains a package (*solve_rec*) whose *solve_rec(recurrence relation, general term, first term)* command can compute the general term of a sequence defined by a recurrence relation. This may be useful when dealing with arithmetic or geometric progressions. Bear in mind that this package must be load previously.

Example: find the general term of an arithmetic progression whose first term is 243 and its difference equals -17.

(%i71) `load("solve_rec")$`

(%i72) `solve_rec(x[n+1]=x[n]-17,x[n],x[1]=243);`

(%o72) $$x_n = 260 - 17\,n$$

10 DERIVATIVES AND OPTIMIZATION

Firstly, let us see how to define functions. Maxima allows us to introduce functions in two different ways, but the assignation operator must be <:=> in both cases. The following examples illustrate this task:

(%i73) `g(x,y):=%e^(x*y)-1/sqrt(x*y);h[x,y,z]:=log(x*y*z);f(x,y):=[1/x,1/y,1/(x*y)];`

(%o73) $$g\,(x,y) := e^{x\,y} - \frac{1}{\sqrt{x\,y}}$$

(%o74) $$h_{x,y,z} := log\,(x\,y\,z)$$

(%o75) $$f\,(x,y) := [\frac{1}{x}, \frac{1}{y}, \frac{1}{x\,y}]$$

In every case, specific values can be obtained by substituting the variables in:

(%i76) `g(-2,-1);h[-3,2,-5];f(3,-4);`

(%o76) $$e^2 - \frac{1}{\sqrt{2}}$$

(%o77) $$log\,(30)$$

(%o78) $$[\frac{1}{3},-\frac{1}{4},-\frac{1}{12}]$$

Besides, we can use *Calculus/Differentiate...* to compute derivatives.

(%i79)
```
'diff(g(x,y),x,2);diff(g(x,y),x,2);
```

(%o79) $$\frac{d^2}{d\,x^2}\left(e^{x\,y}-\frac{1}{\sqrt{x\,y}}\right)$$

(%o80) $$y^2\,e^{x\,y}-\frac{3\,y^2}{4\,(x\,y)^{\frac{5}{2}}}$$

(%i81)
```
diff(h[x,y,z],z,3);
```

(%o81) $$\frac{2}{z^3}$$

(%i82)
```
diff(f(x,y),x,1);
```

(%o82) $$[-\frac{1}{x^2},0,-\frac{1}{x^2\,y}]$$

Maxima contains a *linearalgebra* package, in which two interesting commands are included: *hessian(f,[x1;...;xn])* computes Hessian matrices, and *jacobian([f1,...,fk],[x1,...,xn])* computes Jacobian matrices as well as gradients. The gradient is treated here as a single component vector, which is not so natural. So, we can write some simple sentences to compute gradients and Hessian matrices in the way we usually do.

(%i83)
```
gradient[f,l]:=makelist(diff(f,l[i]),i,1,length(l))$
hessi[f,l]:=makelist(gradient[a,l],a,gradient[f,l])$
hessiana[f,l]:=apply('matrix,hessi[f,l])$
jacobiana[f,l]:=apply('matrix,gradient[f,l])$
```

(%i87)
```
gradient[g(x,y),[x,y]];hessiana[h[x,y,z],[x,y,z]];
jacobiana[f(x,y),[x,y]];
```

(%o87) $$[y\,e^{x\,y}+\frac{y}{2\,(x\,y)^{\frac{3}{2}}},x\,e^{x\,y}+\frac{x}{2\,(x\,y)^{\frac{3}{2}}}]$$

(%o88) $$\begin{pmatrix}-\frac{1}{x^2} & 0 & 0\\ 0 & -\frac{1}{y^2} & 0\\ 0 & 0 & -\frac{1}{z^2}\end{pmatrix}$$

(%o89) $$\begin{pmatrix}-\frac{1}{x^2} & 0 & -\frac{1}{x^2\,y}\\ 0 & -\frac{1}{y^2} & -\frac{1}{x\,y^2}\end{pmatrix}$$

As we have said before, this programming might be a bit hard for students. So we can save our programs in a file that will be treated as a package. For instance, let us save the former orders in a package-file called *mypack.mac* by using the *File/Save as* command and selecting *Maxima batch file* as a type of file. We can supply it to our students and they only need to load it by using the *File/Load package* command and selecting *Name: mypack* from the suitable adress and *Type: All*. As a result, a message like this will be displayed on the screen:

(%i90) `load("C:/Documents and Settings/Mis documentos/Maxima_docs/mypack")$`

10.1 Computing Optima

The theoretical results to compute optima can be used with the aid of the tools we have already seen. Just remark the best way to substitute a vector in a Hessian matrix: by using the *at(matriz hessiana,[v1=k1,...,vn=kn])* command instead of *Simplify/Substitute*.

Example: compute the optima of $f(x,y) = x^4 + y^4 + 2xy$.

(%i6) `/*Critical points*/ gradient[x^4+y^4+2*x*y,[x,y]];`

(%o6) $$[2\,y + 4\,x^3, 4\,y^3 + 2\,x]$$

(%i7) `solve([2*y+4*x^3,4*y^3+2*x],[x,y]);`

(%o7) $[[x = -\frac{i-1}{2}, y = \frac{i+1}{2}], [x = \frac{i+1}{2}, y = -\frac{i-1}{2}], [x = -\frac{1}{\sqrt{2}}, y = \frac{1}{\sqrt{2}}], [x = \frac{1}{\sqrt{2}}, y = -\frac{1}{\sqrt{2}}],$
$[x = -\frac{i+1}{2}, y = \frac{i-1}{2}], [x = \frac{i-1}{2}, y = -\frac{i+1}{2}], [x = -\frac{i}{\sqrt{2}}, y = -\frac{i}{\sqrt{2}}], [x = \frac{i}{\sqrt{2}}, y = \frac{i}{\sqrt{2}}], [x = 0, y = 0]]$

(%i8) `/*Sufficient condition just for (0,0)*/ hessian[x^4+y^4+2*x*y,[x,y]];`

(%o8) $$\begin{pmatrix} 12\,x^2 & 2 \\ 2 & 12\,y^2 \end{pmatrix}$$

(%i9) `/*Substituting*/ at(%,[x=0,y=0]);`

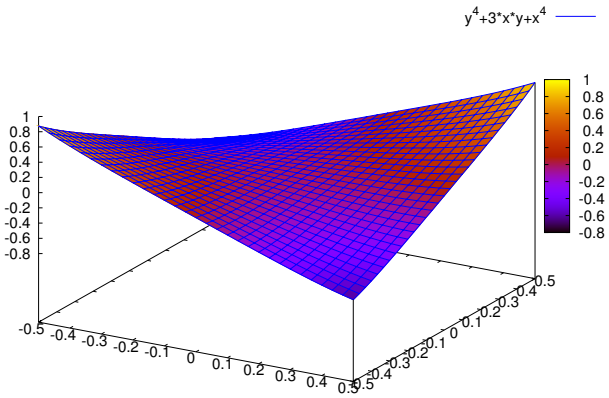
(%o9) $$\begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}$$

(%i10) `/*Eigenvalues*/ eigenvalues(%);`

(%o10) $$[[-2, 2], [1, 1]]$$

So $(0,0)$ is a saddle point. Hence, we have no real optimum this time. In fact, the plotting of the function in a neighborhood of $(0,0)$ is the following:

(%i11) `wxplot3d(x^4+y^4+3*x*y, [x,-0.5,0.5],[y,-0.5,0.5],[plot_format,gnuplot])$`



(%t11)

11 INTEGRATION

When dealing with the integration of real-valued functions, most of common teaching needs are satisfied with the menu *Calculus/Integrate...* Let us see some examples of simple and double integrals.

(%i12) `integrate(x^3-3*x, x);`

(%o12)
$$\frac{x^4}{4}-\frac{3\,x^2}{2}$$

(%i13) `print('integrate(sin(x)-x,x,0,%pi/2),"=",integrate(sin(x)-x,x,0,%pi/2))$`

$$\int_0^{\frac{\pi}{2}} sin\,(x)-x dx=-\frac{\pi^2-8}{8}$$

As an example, let us compute the double integral of the function $f(x,y)=\dfrac{xy}{x^2+y^2}$ on the interval $[2,4]\times[1,3]$ of $\mathbb{R}^2$:

(%i14)
```
'integrate('integrate((x*y)/(x^2+y^2),y,1,3),x,2,4);
integrate(integrate((x*y)/(x^2+y^2), y,1,3),x,2,4);
```

(%o14)
$$\int_2^4 x\int_1^3 \frac{y}{y^2+x^2}dydx$$

(%o15)
$$\frac{25\,log\,(25)}{4}-\frac{17\,log\,(17)}{4}-\frac{13\,log\,(13)}{4}+\frac{5\,log\,(5)}{4}$$

Eventually, Maxima can also compute improper integrals as the following examples illustrate:

```
(%i16) 'integrate(1/x^2,x,1,inf);integrate(1/x^2,x,1,inf);
```

(%o16) $$\int_1^{\infty} \frac{1}{x^2} dx$$

(%o17) 1

This integral is equivalent to:

```
(%i18) limit(integrate(1/x^2,x,1,k),k,inf);
Is  k-1  positive, negative, or zero? positive
```

(%o18) 1

```
(%i19) 'integrate(1/(x^(1/2)),x,0,1);integrate(1/(x^(1/2)),x,0,1);
```

(%o19) $$\int_0^{1} \frac{1}{\sqrt{x}} dx$$

(%o20) 2

And this integral is equivalent to:

```
(%i21) limit(integrate(1/x^(1/2),x,k,1),k,0);
Is  k-1  positive, negative, or zero? negative;
Is  k positive, negative, or zero? positive;
```

(%o21) 2

12 FINAL CONSIDERATIONS

Hopefully, we got a general idea about the possibilities of Maxima in the mathematics classroom from all what has been presented above. But, for the sake of completeness, let us remark some advantages and disadvantages of this software, according to our opinion, of course.

From our point of view, the main advantages of Maxima (under the wxMaxima environment) are:

1. Its workspace is the whole screen, that is, we have at our disposal far more than a single line to introduce expressions.

2. wxMaxima can also be used as a text editor, although a bit limited. The draft of this paper has been elaborated with it.

3. As we have seen before, almost all the educative mathematical needs can be satisfied by using buttons as well as menus, so students hardly need to memorize any command's name.

4. Maxima is a powerful software with very efficient algorithms, as well as a good programming language incorporating, among others procedures, recursive programming.

5. Expressions can be modified or erased in the same place they are on the screen, that is, they do not need to be reedited. Like any text editor, the *cut, copy* and *paste* options are also available.

6. Windows installation is very easy: just, executing the installation file is needed.

7. Maxima is freeware, and all its manuals can be freely distributed. So it is available for any student in any place, even at home. The saved money on license fee can be considerable.

8. This software is being continuously updated (latest version: 5.21.1). The current version provides a huge variety of packages from Discrete Mathematics to Statistics, Numerical Analysis, Geometry and so on. But contributions are always joined to Maxima, getting a better interface and more possibilities, not just for Mathematics (indeed Maxima has a graphical package which allows us to draw a map of most of the countries in the world).

Disadvantages are rather didactic than in terms of efficiency. The most remarkable ones could be:

1. Some kind of data seems to be duplicated or defined in two different ways. This could be caused by the inheritance from Lisp. For instance: we have numerical functions (variables in parenthesis) $f(x, y)$ as well as array or 'memory' functions (variables in brackets) $f[x, y]$; there are two assignation operators: <:> for constants and <:=> for functions; data such as matrices *matrix([],[])* and lists of lists *[[],[]]* are considered to be distinct.

2. Some syntax are not intuitive enough. For instance, we have to write *is(equal(A,B))* instead of *is(A=B)* to check whether A and B are equal.

3. Simplification of numeric expressions is 'compulsory', that is, we cannot see $\frac{\frac{1}{2}+\frac{1}{3}}{2}$ written in this way on screen because $\frac{5}{12}$ (the resulting fraction) is automatically displayed. But this can be avoided with some other symbols like *diff* or *integrate* using the <'> prefix as we have seen before.

4. Despite having pop-up menus, the Maxima version we have used (5.17.1) does not have some palettes of commands to ease its use. These palettes are available for later versions.

However, despite all these disadvantages, we can conclude that wxMaxima is an efficient alternative to the traditional software used in the mathematics classroom.

References

[1] E.M. Fedriani, M.C. Melgar and Á.F. Tenorio. *Matemáticas para la Administración y Dirección de Empresas* Ed. Elaleph.com 2007.

[2] E.M. Fedriani and M.C. Melgar. *Prácticas informáticas para las Matemáticas de LADE.* Dpto. Economía, Métodos Cuantitativos e Historia Económica, Universidad Pablo de Olavide. 2008.

[3] M. Rodríguez and J. Villarte. *Manual de Maxima (versión en español).* maxima.sourceforge.net. 2008.

[4] J.R. Rodríguez Galván. *Maxima con wxMaxima: software libre en el aula de matemáticas.* Dpto. Matemáticas y Oficina de Software Libre de la Univ. de Cádiz. 2007.

[5] M. Rodríguez Riotorto. *Primeros pasos en Maxima.* www.biomates.net. 2006.