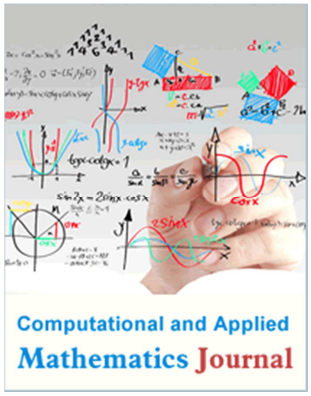
Computational and Applied Mathematics Journal
2015; 1(3): 131-138
Published online April 30, 2015 (http://www.aascit.org/journal/camj)

AASCIT American Association for Science and Technology



Keywords
Symbolic Computations,
Computer-Based Education,
Xcas Computer Software

Received: March 31, 2015
Revised: April 20, 2015
Accepted: April 21, 2015

Using Xcas in Calculus Curricula: a Plan of Lectures and Laboratory Projects

George E. Halkos, Kyriaki D. Tsilika

Laboratory of Operations Research, Department of Economics, University of Thessaly, Volos, Greece

Email address
ktsilika@uth.gr (K. D. Tsilika)

Citation
George E. Halkos, Kyriaki D. Tsilika. Using Xcas in Calculus Curricula: a Plan of Lectures and Laboratory Projects. *Computational and Applied Mathematics Journal.* Vol. 1, No. 3, 2015, pp. 131-138.

Abstract
We introduce a topic in the intersection of symbolic mathematics and computation, concerning topics in multivariable Optimization and Dynamic Analysis. Our computational approach gives emphasis to mathematical methodology and aims at both symbolic and numerical results as implemented by a powerful digital mathematical tool, CAS software Xcas. This work could be used as guidance to develop course contents in advanced calculus curricula, to conduct individual or collaborative projects for programming related objectives, as Xcas is freely available to users and institutions. Furthermore, it could assist educators to reproduce calculus methodologies by generating automatically, in one entry, abstract calculus formulations.

1. Introduction

Educational institutions are equipped with computer labs while modern teaching methods give emphasis in computer based learning, with curricula that include courses supported by the appropriate computer software. It has also been established that, software tools integrate successfully into Mathematics education and are considered essential in teaching Geometry, Statistics or Calculus (see indicatively [1], [2]).

Optimization of multivariable functions ends up with complicated symbolic computations from differential calculus to matrix algebra operations. Dynamic Analysis uses stability conditions also based on symbolics and abstract formulations, as functional matrices and eigen-analysis. In all cases, symbolic and numeric computations can be eliminated using mathematical software.

In the lectures proposed, a solution procedure of multivariable multi equation problems applying - not substituting - theoretical methodology is followed, while, at the same time, the reader is acquainted with a powerful digital mathematical tool, Computer Algebra System (CAS) Xcas1([3]). In this computational approach, calculus methodologies using complex or difficult to remember laws of formation are reproduced. The proposed laboratory projects follow a functional programming approach in Xcas; they succeed to create program files that generate automatically, in one entry, abstract calculus formulations and perform tests in a black box function, in a symbolic language, the Xcas program editor.

In optimization problems, existence theorems for local and global extrema are considered with emphasis in the corresponding necessary and sufficient conditions.

1The selected software, Xcas, is a computer algebra system accessible to all users interested, free of any charges, available at http://www-fourier.ujf-grenoble.fr/~parisse/giac.html. Xcas is compatible with Mac OSX, Windows (except possibly for Vista) and Linux/Ubuntu.

In the dynamic analysis context, existence of equilibrium in dynamic models in discrete and continuous time is checked: stability results of difference equations and ordinary linear and nonlinear differential equations are generated.

The lectures and laboratory projects presented in section 2 can be used in a wide variety of courses in Sciences, Business and Engineering. The solution procedure(s) presented in section 3 provide a general introduction to the problem solving environment Xcas. Section 4 introduces the Xcas program editor. The whole computational approach combines both analytical - theoretical methods and practical programming applications. The last section concludes the paper.

2. Results and Discussion

Once the introduction of the elements of optimization theory and dynamic analysis is made theoretically, the educator may implement selected topics of multivariable calculus with computer-based lectures and laboratory projects as the ones proposed in the present section. All lecture contents are Xcas-based and are aimed to help students, early researchers and scientists who want to perform less and simpler computations compared to working manually. However the students should have an upper level theoretical background in mathematics, specifically in differential and infinitesimal Calculus. No experience of the software package is required. The laboratory projects provide a beginners' introduction to functional programming.

2.1. Lecture 1. Multivariable Unconstrained Optimization Problems

1. Concepts and Operators using built-in Xcas functions: The gradient vector, the Hessian determinant and the leading principal minors of the Hessian determinant

2. Setting and computing necessary conditions for relative extremum in Xcas

3. Setting and computing sufficient conditions for relative extremum in Xcas

4. Example of Application

1st Laboratory Project: Unconstrained Optimization

After introducing the contents of lecture 1, programming the corresponding matrix formulations and the tests required by optimality conditions could be the next learning goal. The Xcas program editor can be applied to create efficient functions for automatic generation of 1) the series of the leading principal minors of Hessian determinant (*hessianseries* function), 2) the signs of the leading principal minors of Hessian determinant evaluated at the critical point(s) (*signserie*s function), in order to test the sufficient conditions for local extrema. The codes to perform such computations can be found in [4] section 4.1.

2.2. Lecture 2. Multivariable Optimization Problems with Equality Constraints

1. Concepts and notions using built-in Xcas functions: the Lagrangian function, the bordered Hessian determinants

2. Setting and computing necessary conditions for relative extremum in Xcas

3. Setting and computing sufficient conditions for relative extremum in Xcas

4. Example of Application

2nd Laboratory Project: Constrained Optimization using classical methods of optimization

After introducing the contents of lecture 2, the corresponding determinental expressions and the tests required by optimality conditions could be programmed. The Xcas program editor can be applied to create efficient functions for automatic generation of 1) the series of bordered Hessian determinants according to sufficient optimality conditions law (*borderhessian* function), 2) the signs of bordered Hessian determinants evaluated at the critical point(s) according to sufficient conditions law (*signborderhessian* function), in order to test the sufficient conditions for local extrema. The codes to perform such computations can be found in [4] section 4.2.

2.3. Lecture 3. Multivariable Optimization Problems with Inequality Constraints

1. Linearly dependent and independent vectors

2. Setting and computing Fritz- John necessary condition in Xcas

3. Setting and computing Kuhn-Tucker Conditions in Xcas

4. Verification of Optimality Conditions at a point in Xcas

3rd Laboratory Project: Constrained Optimization using the mathematical programming approach

The Xcas program editor can be applied to create efficient functions for automatic generation of 1) the expression for Fritz- John necessary condition for local minima (*fritzjohn* function), 2) the expression for Kuhn-Tucker necessary conditions for local minima (*kuhntucker1*function), 3) a test of linear independence of the gradient vectors of the binding constraints (*kuhntucker2* function).The codes to perform such computations can be found in [4] sections 6.1, 6.2.

2.4. Lecture 4. Stability Results for Discrete Time Dynamic Models

1. Concepts and notions using built-in Xcas functions: the Jacobian matrix, eigenvalues and eigenvectors, similarity matrix and the Jordan canonical form of a square matrix

2. Computing the solutions of the characteristic polynomial of the difference equation, the characteristic roots in Xcas

3. Computing determinental expressions to apply Schur's theorem (see [5], [6], [7]) in Xcas

4. Setting and computing necessary and sufficient stability conditions for linear constant coefficient difference equations (as defined in [5], chapter 16) and for systems of linear constant coefficient difference equations (as defined in [5], chapter 18, [8] 5G p.264), in Xcas

5. Example of Application

4rth Laboratory Project: The Xcas program editor can be applied to create efficient functions for automatic generation of 1) the determinant series of the Schur Theorem (*schurseries* function) 2) a direct answer to the stability test for an n-th order linear difference equation based on the Schur Theorem (e.g. stable/unstable) (*stabilitytest1* function), 3) a direct answer to the stability test for systems of linear constant coefficient first order difference equations (e.g. stable/unstable) (*stabilitytest2* function) 4) automatic generation of the asymptotic state of the system in a column matrix form in case equilibrium exists (*steadystate* function). The codes to perform such computations can be found in [9, 10].

2.5. Lecture 5. Stability Results for Continuous Time Dynamic Models

1. Concepts and notions using built-in Xcas functions: the Jacobian matrix
2. Computing determinental expressions to apply Routh's theorem ([11], pp. 429-435) in Xcas
3. Computing the characteristic roots of an n-th order constant coefficient homogeneous linear differential equation and the characteristic equation of a square matrix
4. Setting and computing necessary and sufficient stability conditions for autonomous systems of differential equations (as defined in [8] pp. 275-280, in [12] theorem 8.2.4 and theorems in p.332) and for linear constant coefficient constant term n-th order differential equations (as defined in [12] p. 319), in Xcas
5. Example of Application

5th Laboratory Project: The Xcas program editor can be applied to create efficient functions for automatic generation of 1) routh's determinants symbolically and numerically (*routhseries* and *detrouthseries* functions), 2) a direct test for stability conditions for higher dimensional differential equations by using Routhian analysis and without solving the corresponding characteristic equations (*stabtest* function), 3) a direct test for stability conditions for simultaneous first order linear differential equations with constant coefficients by checking the sign of the real part of the eigenvalues of the coefficient matrix (*linearsystemstability* function), 4) a direct test for stability of equilibrium points for simultaneous nonlinear differential equations, by checking the sign of the real part of the eigenvalues of the jacobian matrix (*nonlinearsystemstability* function). The codes to perform such computations can be found in [10, 13].

3. Teaching Examples2

In this section we review how the software package Xcas performs the computations of content 4 of lectures 1,2,3. The following examples present the computational solution to optimization problems and focus on understanding the underlying theory. A comprehensive presentation supports the lecture learning goals.

2 All computations were made in Xcas version 1.1.2

Computations of content 4 of lectures 4 and 5 could be performed in a similar way.

Example 3.1

Find the extrema of $f(x,y,z)=(x^2+2y^2+3z^2)e^{-(xz^2+y^2+z^2)}$ in the first quadrant of the *xy*-plane (the example is taken from [1]).

We define function *f* in Xcas:

```
f(x,y,z):=(x^2+2*y^2+3*z^2)*exp((-(x*z^2+y^2+z^2)))
```

We first calculate the critical points using the necessary condition (1). In Xcas environment, by writing

```
gradf:=grad(f(x,y,z),[x,y,z])
```

results in:

```
[2*x*exp((-x)*z^2-y^2-z^2)+(x^2+2*y^2+3*z^2)*exp((-x)*z^2-y^2-z^2)*(-(z^2)),

2*2*y*exp((-x)*z^2-y^2-z^2)+(x^2+2*y^2+3*z^2)*exp((-x)*z^2-y^2-z^2)*-2*y,

3*2*z*exp((-x)*z^2-y^2-z^2)+(x^2+2*y^2+3*z^2)*exp((-x)*z^2-y^2-z^2)*((-x)*2*z-2*z)]
```

The necessary condition based on the gradient test is not solvable due to the exponential factors. After factorization of the components of the gradient vector gradf:

```
factor(gradf[[1]]);factor(gradf[[2]]);factor(gradf[[3]]);
```

```
((-(x^2))*z^2+2*x-2*y^2*z^2-3*z^4)*exp((-x)*z^2-y^2-z^2), -2*y*(x^2+2*y^2+3*z^2-2)*exp((-x)*z^2-y^2-z^2), -2*z*(2*y^2*x+2*y^2+x^3+x^2+3*x*z^2+3*z^2-3)*exp((-x)*z^2-y^2-z^2)
```

Solution is accomplished using the following simplified form of the equations:

```
solution:=solve([(-x^2*z^2+2*x-2*y^2*z^2-3*z^4)=0,(-2*y)*(x^2+2*y^2+3*z^2-2)=0,(-2*z)*(2*y^2*x+2*y^2+x^3+x^2+3*x*z^2+3*z^2-3)=0],[x,y,z])
```

```
[[0.538962270589,-0.0,0.743613967821],[0.538962270589,0.0,-0.743613967821], [1/2,(2*sqrt(2))/8,(sqrt(2))/2], [1/2,-((2*sqrt(2))/8),(sqrt(2))/2], [1/2,(2*sqrt(2))/8,-((sqrt(2))/2)], [1/2,-((2*sqrt(2))/8),-((sqrt(2))/2)], [0,1,0], [0,-1,0], [0,0,0]]
```

We get nine critical points. The nonnegative ones are selected:

```
select(solution->sign(solution)==[1,1,1] or sign(solution)=[1,0,0] or sign(solution)=[1,1,0] or sign(solution)=[0,1,1] or sign(solution)=[0,0,0] or sign(solution)=[1,0,1] or sign(solution)==[0,1,0],solution)
```

```
[[0.538962270589,-0.0,0.743613967821],[1/2,(sqrt(2))/4,(sqrt(2))/2],[0,1,0],[0,0,0]]
```

The nonnegative critical points as components of the initial solution are denoted by:

solution[[1]];solution[[3]];solution[[7]];solution[[9]]

[0.538962270589,-0.0,0.743613967821],[1/2,(sqrt(2))/4,(sqrt(2))/2],[0,1,0],[0,0,0]

Hessian matrix is:

hessian(f(x,y,z),[x,y,z])

$$\begin{bmatrix} 2\,e^{-x^2z^2-y^2-z^2}-4\,x^2z^2\,e^{-x^2z^2-y^2-z^2}+z^4\,e^{-x^2z^2-y^2-z^2}(x^2+2y^2+3z^2) & -4\,xy\,e^{-x^2z^2-y^2-z^2}-4\,yz^2\,e^{-x^2z^2-y^2-z^2}+2yz^2\,e^{-x^2z^2-y^2-z^2}(x^2+2y^2+3z^2) & -6z^3\,e^{-x^2z^2-y^2-z^2}+2x^2z\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)-2z\,e^{-x^2z^2-y^2-z^2}(x^2+2y^2+3z^2)-z^2\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)(x^2+2y^2+3z^2) \\ -4\,xy\,e^{-x^2z^2-y^2-z^2}-4\,yz^2\,e^{-x^2z^2-y^2-z^2}+2yz^2\,e^{-x^2z^2-y^2-z^2}(x^2+2y^2+3z^2) & 4\,e^{-x^2z^2-y^2-z^2}-16y^2\,e^{-x^2z^2-y^2-z^2}-2\,e^{-x^2z^2-y^2-z^2}(x^2+2y^2+3z^2)+4y^2\,e^{-x^2z^2-y^2-z^2}(x^2+2y^2+3z^2) & -12yz\,e^{-x^2z^2-y^2-z^2}+4yz\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)-2yz\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)(x^2+2y^2+3z^2) \\ -6z^3\,e^{-x^2z^2-y^2-z^2}+2x^2z\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)-2z\,e^{-x^2z^2-y^2-z^2}(x^2+2y^2+3z^2)-z^2\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)(x^2+2y^2+3z^2) & -12yz\,e^{-x^2z^2-y^2-z^2}+4yz\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)-2yz\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)(x^2+2y^2+3z^2) & 6\,e^{-x^2z^2-y^2-z^2}+12z^2\,e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)+e^{-x^2z^2-y^2-z^2}(-2x^2z-2z)^2(x^2+2y^2+3z^2)+e^{-x^2z^2-y^2-z^2}(-2x^2-2)(x^2+2y^2+3z^2) \end{bmatrix}$$

Next we check the sequence of the leading principal minors of the Hessian determinant, evaluated at every nonnegative critical point:

map([1,3,7,9],j->seq(approx(subst(det(hessian(f(x,y,z),[x,y,z])[0..k,0..k]), [x,y,z]=solution[[j]])), k=0.length(hessian(f(x,y,z),[x,y,z]))-1))

[[0.599476634511,0.0259220576912,-0.340062512248], [0.625293029518,-0.347547886901,0.579518056274], [0.735758882343,-2.16536453179,-1.59318618777], [2.0,8.0,48.0]]

Alternatively, only the signs of the leading principal minors of Hessian determinant are evaluated at every nonnegative critical point:

map([1,3,7,9], j->seq(sign(subst(det(hessian(f(x,y,z),[x,y,z])[0..k,0..k]), [x,y,z]=solution[[j]])), k=0..length(hessian(f(x,y,z),[x,y,z]))-1))

[[1,1,-1],[1,-1,1],[1,-1,-1],[1,1,1]]

Note: 1 stands for sign (+) and -1 stands for sign (-)

Results: the critical point (0,0,0) (or for Xcas solution[[9]]) satisfies necessary and sufficient optimality conditions for relative minimum.

Example 3.2

Find the relative extrema of the function $f(x,y,z,w) = 0.3\log(x-2) + 0.4\log(y-3) + 0.2\log(z-4) + 0.1\log(w-5)$, subject to the constraint $g(x,y,z,w) = 2x + 3y + 4z + 5w - 100 = 0$ (the example is taken from [1]).

We define the Lagrangian function

l:=0.3*log(x-2)+0.4*log(y-3)+0.2*log(z-4)+0.1*log(w-5)+λ*(2*x+3*y+4*z+5*w-100)

We calculate a critical point of the Lagrangian function:

criticalpoint:=solve(grad(l,[x,y,z,w,λ])=[0,0,0,0,0],[x,y,z,w,λ])

[[8.9,9.13333333333,6.3,5.92,-0.0217391304348]]

We compute the leading principal minors of the bordered Hessian determinant:

seq(hessian(l,[λ,x,y,z,w])[0..t,0..t],t=2..length(hessian(l,[λ,x,y,z,w]))-1)

$$\left(\begin{bmatrix} 0 & 2 & 3 \\ 2 & \frac{0.3\cdot(-1)}{(x-2)^2} & 0 \\ 3 & 0 & \frac{0.4\cdot(-1)}{(y-3)^2} \end{bmatrix}, \begin{bmatrix} 0 & 2 & 3 & 4 \\ 2 & \frac{0.3\cdot(-1)}{(x-2)^2} & 0 & 0 \\ 3 & 0 & \frac{0.4\cdot(-1)}{(y-3)^2} & 0 \\ 4 & 0 & 0 & \frac{0.2\cdot(-1)}{(z-4)^2} \end{bmatrix}, \begin{bmatrix} 0 & 2 & 3 & 4 & 5 \\ 2 & \frac{0.3\cdot(-1)}{(x-2)^2} & 0 & 0 & 0 \\ 3 & 0 & \frac{0.4\cdot(-1)}{(y-3)^2} & 0 & 0 \\ 4 & 0 & 0 & \frac{0.2\cdot(-1)}{(z-4)^2} & 0 \\ 5 & 0 & 0 & 0 & \frac{0.1\cdot(-1)}{(w-5)^2} \end{bmatrix}\right)$$

Previous result evaluated at the critical point:

subst([seq(det(hessian(l,[λ,x,y,z,w])[0..t,0..t]),t=2..length(hessian(l,[λ,x,y,z,w]))-1)],[x,y,z,w,λ]=criticalpoint[[1]])

[0.0992443186526,-0.00482419714138,0.000633296507025]

Alternatively, only the determinants' signs at the critical point are evaluated:

subst([seq(sign(det(hessian(l,[λ,x,y,z,w])[0..t,0..t])),t=2..length(hessian(l,[λ,x,y,z,w])) 1)],[x,y,z,w,λ]=criticalpoint[[1]])

[1.0,-1.0,1.0]

Sufficient optimality conditions for the critical point are satisfied for relative maximum.

Example 3.3

Consider the constrained optimization problem (the example is taken from [1])

$\min u(x,y,z,w) = x^2 + 2y^2 + 2z^2 + w^2 - 2xy - 2yz - 2zw$

subject to the constraints:

$$g^1(x,y,z,w) = 2x + y + z + w - 10 = 0, g^2(x,y,z,w) = 4x - z + w - 12 = 0$$

In Xcas environment we define the Lagrangian function

u(x,y,z,w,λ,μ):=x^2+2*y^2+2*z^2+w^2-2*x*y-2*y*z-2*z*w+λ*(2*x+y+z+w-10)+μ*(4*x-z+w-12)

seq(hessian(u(x,y,z,w,λ,μ),[λ,μ,x,y,w,z])[0..t,0..t],t=4..length(hessian(u(x,y,z,w,λ,μ),[λ,μ,x,y,w,z]))-1)

$$\left(\begin{bmatrix} 0 & 0 & 2 & 1 & 1 \\ 0 & 0 & 4 & 0 & 1 \\ 2 & 4 & 2 & -2 & 0 \\ 1 & 0 & -2 & 4 & 0 \\ 1 & 1 & 0 & 0 & 2 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 2 & 1 & 1 & 1 \\ 0 & 0 & 4 & 0 & 1 & -1 \\ 2 & 4 & 2 & -2 & 0 & 0 \\ 1 & 0 & -2 & 4 & 0 & -2 \\ 1 & 1 & 0 & 0 & 2 & -2 \\ 1 & -1 & 0 & -2 & -2 & 4 \end{bmatrix}\right)$$

Evaluating determinant's values:

seq(det(hessian(u(x,y,z,w,λ,μ),[λ,μ,x,y,w,z])[0..t,0..t]),t=4..length(hessian(u(x,y,z,w,λ,μ),[λ,μ,x,y,w,z]))-1)

42,836

Or determinant's signs:

seq(sign(det(hessian(u(x,y,z,w,λ,μ),[λ,μ,x,y,w,z])[0..t,0..t])),t=4..length(hessian(u(x,y,z,w,λ,μ),[λ,μ,x,y,w,z]))-1)

1,1

Sufficient optimality conditions are satisfied for relative minimum.

Example 3.4

Consider the optimization problem:

minimize $f(x,y) = (x-8)^2 + (y-6)^2$ subject to $g_1(x,y) = x + y - 7 \le 0, g_2(x,y) = 4x + y - 16 \le 0, \quad g_3(x,y) = -x + y - 5 \le 0,$ $x \ge 0, y \ge 0$ Check whether the points (3,4) and (1,6) satisfy the Fritz – John necessary condition (the example is taken from [1]).

In Xcas environment we define

f(x,y):=(x-8)^2+(y-6)^2;g1(x,y):=x+y-7;g2(x,y):=4*x+y-16;g3(x,y):=-x+y-5;

We search for binding constraints at (3,4) and (1,6):

g1(3,4);g2(3,4);g3(3,4)

0,0,-4

(constraints g1, g2 are binding since g1(3,4)=0=g2(3,4) and g3 is not since g3(3,4)≠0)

g1(1,6);g2(1,6);g3(1,6)

0,-6,0

(constraints g1, g3 are binding since g1(1,6)=0=g3(1,6) and g2 is not since g2(1,6)≠0)

Substituting point (3,4) in the first part of the Fritz-John condition:

subst(λ0*grad(f(x,y),[x,y])+λ1*grad(g1(x,y),[x,y])+λ2*grad(g2(x,y),[x,y]),[x,y]=[3,4])

[-10*λ0+λ1+4*λ2,-4*λ0+λ1+λ2]

solve([-10*λ0+λ1+4*λ2=0,-4*λ0+λ1+λ2=0],[λ1,λ2])

[[2*λ0,2*λ0]] (we prove the existence of parameters λ0, λ1, λ2≥0 as Fritz-John necessary condition assumes for local minimum)

Substituting point (1,6) in the first part of the Fritz-John condition:

subst(λ0*grad(f(x,y),[x,y])+λ1*grad(g1(x,y),[x,y])+λ3*grad(g3(x,y),[x,y]),[x,y]=[1,6])

[-14*λ0+λ1-λ3,λ1+λ3]

solve([-14*λ0+λ1-λ3=0,λ1+λ3=0],[λ1,λ3])

[[7*λ0,-7*λ0]] (we prove that there are no parameters λ0, λ1, λ3≥0 as Fritz-John necessary condition assumes for local minimum)

4. Programming in Xcas

Programs may be written in a command line if they are one or two lines long, but for more complex programs, it is a good idea to put them in a separate program level, via Prg->New of Prg Menu. This will open an editor in a new level. The editor has its own menu, where we can open or import an existing (program) file inside the current text, save or export the current program. We can also use the editor menu to insert programming structure. There are buttons to find the next occurrence of a search string, to parse the current program (errors are displayed in the messages area) and to save the program (the current filename is displayed at the right of the save button). The OK button (F9) is used for compilation ([3]).

In this section we illustrate the realization of the 5th Laboratory Project in Xcas. The codes for *routhseries* *detrouthseries* and *stabtest* functions (see [10], [13]) are saved in rouththeorem.cxx program file. The codes for *linearsystemstability* and *nonlinearsystemstability* functions (see [10], [13]) are saved in eigentest.cxx program file.

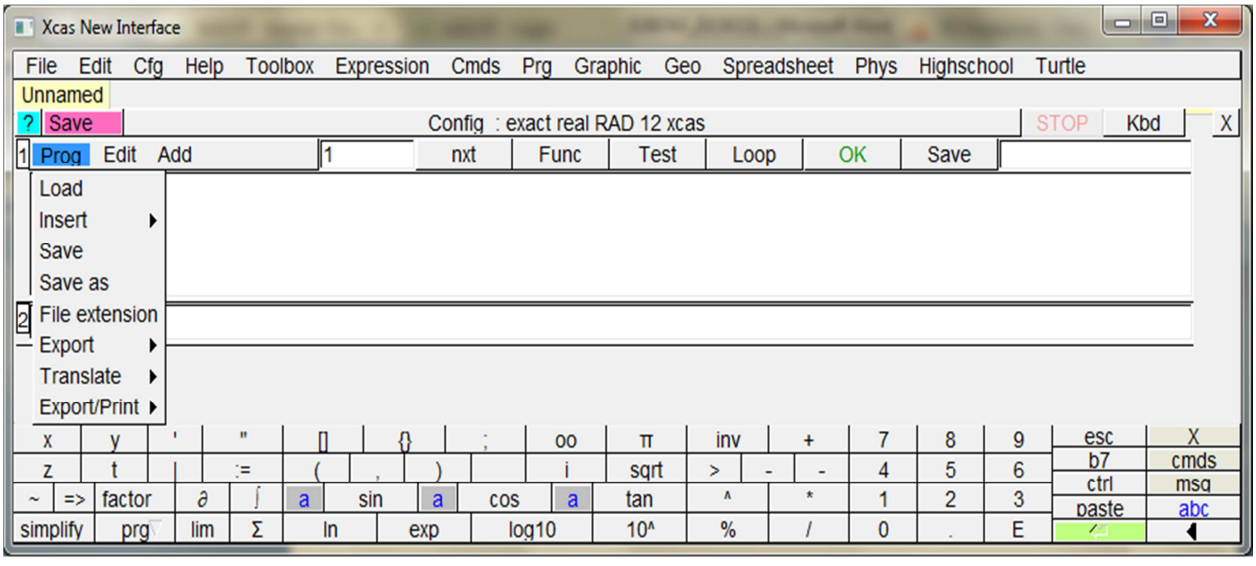


Figure 1. *Loading a program file in Xcas.*

Example 4.1

Working in any session in Xcas, by writing in a command line read("rouththeorem.cxx") we can use *routhseries*, *detrouthseries* and *stabtest* functions.

routhseries function takes as arguments the characteristic polynomial of the linear differential equation and its variable. *routhseries* generates the first n minors of determinant $\Delta_n = \begin{vmatrix} a_1 & a_0 & 0 & 0 & \dots & 0 \\ a_3 & a_2 & a_1 & a_0 & \dots & 0 \\ a_5 & a_4 & a_3 & a_2 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_n & a_{n-1} \\ 0 & 0 & 0 & 0 & \dots & a_n \end{vmatrix}$. To calculate the first n minors of Δ_n we use *detrouthseries* function with arguments the characteristic polynomial and its variable.

In Xcas environment, by writing read ("rouththeorem.cxx") we can generate the input of our programmed functions as presented below:

rout series (a0*x^5+a1*x^4+a2*x^3+a3*x^2+a4*x+a5,x)

$$\left([a1], \begin{bmatrix} a1, & a0 \\ a3, & a2 \end{bmatrix}, \begin{bmatrix} a1, & a0, & 0 \\ a3, & a2, & a1 \\ a5, & a4, & a3 \end{bmatrix}, \begin{bmatrix} a1, & a0, & 0, & 0 \\ a3, & a2, & a1, & a0 \\ a5, & a4, & a3, & a2 \\ 0, & 0, & a5, & a4 \end{bmatrix}, \begin{bmatrix} a1, & a0, & 0, & 0, & 0 \\ a3, & a2, & a1, & a0, & 0 \\ a5, & a4, & a3, & a2, & a1 \\ 0, & 0, & a5, & a4, & a3 \\ 0, & 0, & 0, & 0, & a5 \end{bmatrix} \right)$$

routhseries(x^3+6*x^2+11*x+6,x)

$$\left([6], \begin{bmatrix} 6, & 1 \\ 6, & 11 \end{bmatrix}, \begin{bmatrix} 6, & 1, & 0 \\ 6, & 11, & 6 \\ 0, & 0, & 6 \end{bmatrix} \right)$$

detrouthseries(x^3+6*x^2+11*x+6,x)
6,60,360

linearsystemstability and *nonlinearsystemstability* functions (see [10], [13]) are saved in eigentest.cxx program file and can be used in any session by writing read("eigentest.cxx"). *linearsystemstability* function takes system's coefficient matrix as argument and returns «asymptotically stable» for systems with equilibrium state(s) and «unstable» for systems that have explosive behavior otherwise *nonlinearsystemstability* function takes as arguments the list of functions of the second part of the d.e. system $\dot{x} = f(x)$, $f = (f_1,...,f_n)^T$, the variable vector and the equilibrium point. *nonlinearsystemstability* function returns «asymptotically stable» for systems with equilibrium state(s) and «unstable» otherwise.

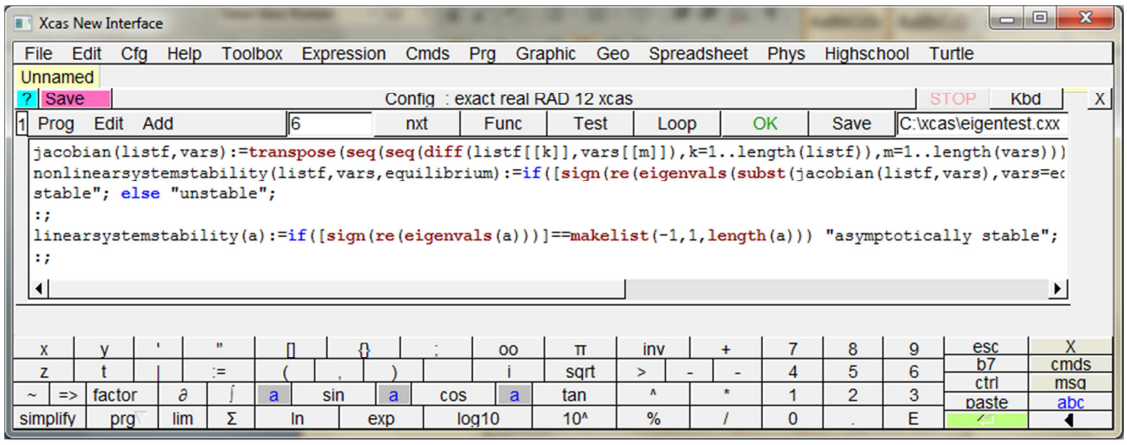


Figure 2. *The eigentest.cxx program file.*

5. Conclusions

The lectures proposed provide thematic applications of the free computer algebra system Xcas to aid instruction of advanced topics in Symbolic Mathematics. The essential idea is to utilize all calculation modules to solve multivariable problems of computational nature. The solution procedure(s) presented can be used in a wide variety of courses in Sciences, Business and Engineering.

The accompanied laboratory projects provide a general introduction to a symbolic language, the Xcas program editor. The lectures combine both analytical theoretical methods and practical programming applications. The importance of calculus methodology within an "all in" general purpose computing environment was highlighted.

References

[1] C. Huang and P. Crooke, *Mathematics and Mathematica for Economists*, Oxford Blackwell Publishers, Massachusetts, 1997.

[2] S. Kadry, M. El Shalkamyb, Toward New Vision in Teaching Calculus, *IERI Procedia*, vol. 2, pp. 548–553, 2012.

[3] B. Parisse, *An Introduction to the Xcas Interface*. Available at http://www-fourier.ujf-grenoble.fr/~parisse/giac/tutoriel_en.pdf

[4] G. E. Halkos and K. D. Tsilika, Computing Optimality Conditions in Economic Problems, *Journal of Computational Optimization in Economics and Finance*, vol. 3, no. 3, pp. 143-155, 2011.

[5] A. Chiang, *Fundamental Methods of Mathematical Economics,* 3 Edn., McGraw-Hill Book, Singapore, 1984.

[6] E.I. Jury, *Inners and Stability of Dynamic Systems*, Wiley, New York, 1974.

[7] M. Neumann, Weak stability for matrices, *Linear Multilinear Algebra,* vol. 7, pp. 257-262, 1979.

[8] G. Strang, Linear *Algebra and its Applications*, 3rd Edn., Harcourt Brace Jovanovich College, Philadelphia, New York, 1988.

[9] G. E. Halkos and K. D. Tsilika, Computational Techniques for Stability Analysis of a Class of Discrete Time Discrete State Dynamic Economic Models, *American Journal of Applied Sciences*, vol. 9, no. 12, pp. 1944-1952, 2012. DOI: 10.3844/ajassp.2012.1944.1952

[10] G. E. Halkos and K. D. Tsilika, Stability Analysis in Economic Dynamics: A Computational Approach, MPRA paper, 2012. Available at http://mpra.ub.uni-muenchen.de/41371/

[11] P.A. Samuelson, *Foundations of Economic Analysis*, Harvard University Press, 1947.

[12] W.B. Zhang, *Differential Equations, Bifurcations, and Chaos in Economics*. Series on Advances in Mathematics for Applied Sciences Vol. 68, World Scientific, New Jersey, 2005.

[13] G. E. Halkos and K. D. Tsilika, Xcas as a Programming Environment for Stability Conditions of a Class of Linear Differential Equation Models in Economics, *AIP Conference Proceedings 1389, 9th International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2011)*, Halkidiki, Greece, 2011, pp. 1769-1772. DOI:10.1063/1.3636951.