Constructing a Generator of Matrices with Pattern

Halkos, George and Tsilika, Kyriaki

University of Thessaly, Department of Economics

2012

Constructing a Generator of Matrices with Pattern

George E. Halkos and Kyriaki D. Tsilika
Laboratory of Operations Research
Department of Economics, University of Thessaly,
Korai 43, 38 333, Volos, Greece

Abstract
Computations with large matrices work out faster with computer software, even faster creating automatically the matrix of the size and pattern needed. In this paper we propose free computer algebra system Xcas resources to display particular matrices that can be called up directly. Our computer codes provide shortcuts for entering random block diagonal matrices, random triangular matrices, random and specialized band matrices, elementary matrices E_{ij}, Fourier matrices. As for matrices needed in the study of mathematical issues concerning the properties of the roots of a polynomial, we create features with polynomial coefficients. We also propose codes for immediate construction of functional matrices such as Jacobian, bordered Hessian and Wronskian. The computer codes proposed provide visual representation of the matrix pattern (which is traditionally explained using indices and numerals), infinite number of examples using random numbers and immediate construction of large matrices of various forms.

Keywords: Matrices with pattern; functional programming; computer software.

JEL Classification Codes: C63; C02; C88; C62.

An earlier version of this paper has been published in *International Journal of Information Science and Computer Mathematics* **4(2):** 101-117, 2011. The present paper includes the upgraded version of the computer codes, which are written in Xcas 0.9.9.

1. Introduction

Matrices with pattern have a wide range of applications in research areas of Bioinformatics (see e.g. Heppell et al., 2000; Hertz et al., 1990), Linear Algebra (see e.g. Stadelmaier et al., 1982; Elsner and Johnson, 1989, Johnson, 1983; Hall and Wang, 2001; McDonald et al., 1997; Tardos, 2005), Structural Mechanics (Kaven and Sayarinejad, 2004), Economics (Cassetti, 1995; Veinott, 1969; Tarr, 1976). In biology, Gene matrices have a 0-1 structure. For models described by linear systems of equations with recursive and block recursive structure, matrices with pattern have a role to play.

In Econometrics, among the full structural simultaneous equation models, the model developed by Wold (1954) is known as a recursive system. In simple recursive systems the coefficient matrix of the jointly dependent variables is triangular and the covariance matrix is diagonal. In cases where the whole system of simultaneous equations decomposes into recursive subsystems, block recursive systems are formulated. Block recursive systems (Lloyd and Lee, 1976; Wermuth, 1992) compared to simple recursive systems, allow important simplifications in the estimation process. Then, the coefficient matrix of the system's jointly dependent variables is block triangular and the covariance matrix of the error terms is block diagonal.

The family of classical interregional input-output models may be classified and compared in terms of the assumed structure of their corresponding matrix of interregional trade share coefficients (Batten and Martellato, 1985). Matrices with pattern are especially useful in the study of dynamic discrete time economic models and dynamic Leontief models.

A mathematical software is equipped with a collection of built-in functions for immediate construction of several matrix families either elementary like zeroes or ones, identity, symmetric, random or general, diagonal, general band or specialized like positive definite, positive definite band, symmetric indefinite, Hermitian indefinite, triangular, general tridiagonal, positive definite tridiagonal, Vandermonde, Hessenberg, Hadamard, Hankel, Hilbert, Pascal, Toeplitz (for the related matrix theory see Strang, 1988; Anton, 2000; Goldberg, 1991 and Lipschutz, 1987).

A brief overview of computer software capabilities in matrix creation, results in various different choices. MATLAB, the most efficient tool in matrix computation, has the largest collection of special matrices. The gallery function in MATLAB holds over fifty different test matrix functions (Quarteroni and Saleri, 2006). Computer algebra systems like *Mathematica,* wxMaxima and Xcas have also matrix functions to return highly specialized matrices, including common functional matrices and coefficient matrices (Anton et al., 2003; Parisse[1]). The contribution of Linear Algebra package, performing exclusively linear algebra operations, is limited in constructing elementary matrices.

Computer codes or matrix functions for the construction of matrices with complex law of formation are not available in commonly used computer software. Then, a user should have programming skills to get the desired results.

In this paper, free computer algebra system Xcas is used to construct a matrix generator, programmed in Xcas program editor. Our matrix generator program file has a number of specialized matrix functions that create different kinds of matrices

[1] Xcas is a Computer Algebra System available free in http://www-fourier.ujf-grenoble.fr/~parisse /giac.html

not included in typical computer algebra software. Potential usefulness of our matrix generator is for

i) verification of algebraic properties and behavior of matrices of special forms i.e. the inverse of a bidiagonal matrix is lower triangular, the inverse of a band matrix is a full matrix, the Fibonacci determinants follow the formula $|F_n| = |F_{n-1}| + |F_{n-2}|$ etc.

ii) technology applications using an interactive software tool

iii) creation of the matrix needed, simplifying and abbreviating the law of formation

iv) guidance for the user to program more matrix functions.

Conclusively, this paper gives the computer codes for several matrix families, describes their input and gives examples of their use. The structure of the paper is the following. Section 2 presents Computer Algebra System Xcas and discusses briefly the programming commands in Xcas. Section 3 presents the codes for automatic generation of block diagonal and triangular matrices using entries of a random number generator, for automatic generation of random and special band matrices and of Fourier matrices, for elementary matrices, for coefficient matrices related to traditional Algebra theorems, for specific functional matrices. The last section concludes the paper.

2. The Computer Algebra System Xcas

2.1 The Xcas system

Xcas is a Computer Algebra System, (CAS), which was developed by Bernard Parisse, at the University of Grenoble, France. In addition to its algebraic capabilities

Xcas incorporates a Dynamical Geometry System, (DGS), in two and three dimensions, spreadsheets, and programming both in a Logo-like language and in its own language. Justifiably, it has been called the “swiss knife for mathematics”. Xcas is a free system available for Mac OS X, Windows (except possibly for Vista) and Linux/Ubuntu; in the File menu it contains an option available for the automatic update of the system. The on-line help is easily accessible and provides numerous examples of each function. Moreover, a user's manual is available, which proves very helpful. Xcas has been translated in several languages. In several localizations there is a users' forum available.

2.2 Programming in Xcas[2]

Programs in Xcas may be written 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 import our computer codes of the following section, separated by «**:;**», save and export the current program as matrix generator.cxx. Working in any session of Xcas, by writing in a commandline read("matrix generator.cxx") we can use multiblockdiagonal, uppertriang, uppertriang2, lowertriang, lowertriang2, bandmatrix, tridiagonal, bidiagonal, fibonacci, fourier, elementary, schur, routh, jacobian, borderhessian, wronskian functions.

3. Construction of Matrices

3.1 Random Block Diagonal Matrices

Our programmed function multiblockdiagonal(m,n) generates square matrices with n m×m random blocks along the main diagonal and zeros everywhere else:

[2] The present paper includes computer codes written in Xcas 0.9.9.

multiblockdiagonal(m,n):=BlockDiagonal([[seq(randmatrix(m,m),n)]])

For example, by writing in Xcas:

multiblockdiagonal(3,5), the output is:

$$\left[\begin{array}{ccccccccccccccc}
85, & 60, & -96, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
-14, & 18, & 92, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
23, & -27, & -88, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 85, & 60, & -96, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & -14, & 18, & 92, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 23, & -27, & -88, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 85, & 60, & -96, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & -14, & 18, & 92, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 23, & -27, & -88, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 85, & 60, & -96, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & -14, & 18, & 92, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 23, & -27, & -88, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 85, & 60, & -96 \\
0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & -14, & 18, & 92 \\
0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 23, & -27, & -88
\end{array}\right]$$

Using built-in function BlockDiagonal(Lst(l)||Mtrx(A)) we can create random blocks of any dimension along the main diagonal, by writing in Xcas:

BlockDiagonal([randmatrix(2,2),randmatrix(3,3),randmatrix(5,5)])

the output is the following:

$$\left[\begin{array}{cccccccccc}
-85, & -44, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
-34, & 5, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 49, & 25, & -71, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 64, & -74, & -27, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 72, & 38, & -88, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & 0, & 0, & 0, & 62, & 21, & 37, & -16, & 8 \\
0, & 0, & 0, & 0, & 0, & 66, & -54, & -15, & -64, & -17 \\
0, & 0, & 0, & 0, & 0, & -81, & 62, & 73, & -75, & 15 \\
0, & 0, & 0, & 0, & 0, & -95, & 77, & -71, & -81, & -67 \\
0, & 0, & 0, & 0, & 0, & -6, & 24, & 81, & -80, & 52
\end{array}\right]$$

Random blocks in blockdiagonal matrices built, are generated by Xcas function randmatrix(n,n), which returns a matrix of size n×m containing random integers.

3.2 Random Triangular and Strictly Triangular Matrices

Our programmed function uppertriang(m) generates random upper triangular square matrices of size m:

uppertriang(m):=matrix(m,m,(j,k)->if(j>k) 0;else rand(1000);)

For example, by writing in Xcas:

uppertriang(7), the output is:

$$\begin{bmatrix} 622, & 301, & 607, & 939, & 914, & 141, & 233 \\ 0, & 873, & 921, & 202, & 407, & 590, & 955 \\ 0, & 0, & 903, & 318, & 748, & 983, & 340 \\ 0, & 0, & 0, & 673, & 788, & 737, & 109 \\ 0, & 0, & 0, & 0, & 463, & 883, & 502 \\ 0, & 0, & 0, & 0, & 0, & 826, & 159 \\ 0, & 0, & 0, & 0, & 0, & 0, & 874 \end{bmatrix}$$

Our programmed function uppertriang2(m) generates random strictly upper triangular square matrices of size m:

uppertriang2(m):=matrix(m,m,(j,k)->if(j>=k)0;else rand(1000);)

For example, by writing in Xcas:

uppertriang2(6), the output is

$$\begin{bmatrix} 0, & 159, & 258, & 442, & 223, & 5 \\ 0, & 0, & 621, & 920, & 376, & 291 \\ 0, & 0, & 0, & 788, & 707, & 633 \\ 0, & 0, & 0, & 0, & 521, & 10 \\ 0, & 0, & 0, & 0, & 0, & 6 \\ 0, & 0, & 0, & 0, & 0, & 0 \end{bmatrix}$$

Accordingly, our programmed functions lowertriang(m) and lowertriang2(m) generate random lower and strictly lower triangular square matrices of size m:

lowertriang(m):=matrix(m,m,(j,k)->if(j<k) 0;else rand(1000);)

lowertriang2(m):=matrix(m,m,(j,k)->if(j<=k)0;else rand(1000);)

For example, by writing in Xcas:

lowertriang(14), the output is

$$\begin{bmatrix} 516, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ 847, & 281, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ 6, & 289, & 72, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ 671, & 9, & 571, & 566, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ 946, & 582, & 731, & 781, & 469, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ 12, & 935, & 807, & 849, & 300, & 364, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ 698, & 230, & 645, & 107, & 385, & 167, & 255, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ 623, & 235, & 94, & 139, & 82, & 376, & 146, & 372, & 0, & 0, & 0, & 0, & 0, & 0 \\ 448, & 817, & 381, & 20, & 384, & 328, & 602, & 116, & 110, & 0, & 0, & 0, & 0, & 0 \\ 72, & 128, & 45, & 880, & 977, & 345, & 245, & 676, & 575, & 890, & 0, & 0, & 0, & 0 \\ 783, & 961, & 58, & 39, & 584, & 293, & 133, & 724, & 376, & 509, & 870, & 0, & 0, & 0 \\ 748, & 958, & 687, & 129, & 978, & 71, & 457, & 581, & 188, & 567, & 654, & 316, & 0, & 0 \\ 613, & 534, & 293, & 958, & 779, & 969, & 534, & 669, & 753, & 495, & 727, & 792, & 80, & 0 \\ 21, & 925, & 804, & 397, & 435, & 675, & 145, & 393, & 362, & 275, & 372, & 434, & 733, & 953 \end{bmatrix}$$

By writing in Xcas:

lowertriang2(6), the output is

$$\begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0 \\ 560, & 0, & 0, & 0, & 0, & 0 \\ 446, & 510, & 0, & 0, & 0, & 0 \\ 44, & 749, & 856, & 0, & 0, & 0 \\ 948, & 75, & 517, & 251, & 0, & 0 \\ 46, & 897, & 156, & 277, & 217, & 0 \end{bmatrix}$$

In the examples above nonzero entries are non-negative 3-digit random integers generated by Xcas function rand(1000).

3.3 Random Band Matrices

Let us define a band matrix as a matrix with zero entries except within the band $|i-j| \le w$. Our programmed function bandmatrix(n,w) generates matrices of size n with w nonzero entries above and below the principal diagonal:

```
bandmatrix(n,w):=matrix(n,n,(i,j)->if(abs(i-j)<=w)rand(-10..10)(); else  0;)
```

For example, by writing in Xcas:

bandmatrix(10,3), the output is

$$\begin{bmatrix} -2.704 & -7.09 & -2.078 & -3.621 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.8127 & -2.607 & 6.605 & 4.443 & -5.734 & 0 & 0 & 0 & 0 & 0 \\ -5.602 & -6.404 & 2.512 & 7.87 & 3.036 & 4.958 & 0 & 0 & 0 & 0 \\ -4.38 & 9.682 & -5.352 & -9.255 & -2.622 & 1.13 & 3.062 & 0 & 0 & 0 \\ 0 & -9.366 & -6.45 & 1.599 & 9.371 & -5.056 & 9.659 & 9.775 & 0 & 0 \\ 0 & 0 & 0.7385 & 7.555 & -2.929 & 3.649 & -4.523 & 3.45 & -7.164 & 0 \\ 0 & 0 & 0 & 2.87 & 0.05427 & 7.279 & 7.136 & 4.452 & -9.125 & -0.3515 \\ 0 & 0 & 0 & 0 & 2.323 & 3.911 & -5.393 & 7.943 & 3.593 & -0.7458 \\ 0 & 0 & 0 & 0 & 0 & 8.687 & -9.03 & -9.616 & 1.75 & -8.395 \\ 0 & 0 & 0 & 0 & 0 & 0 & -6.066 & -6.651 & -9.025 & -1.122 \end{bmatrix}$$

The following codes create tridiagonal and bidiagonal matrices of size n:

```
tridiagonal(n):=matrix(n,n,(i,j)->if(abs(i-j)<=1)rand(-10..10)(); else  0; )
```

For example, by writing in Xcas:

tridiagonal(7), the output is:

$$\begin{bmatrix} 9.083 & -3.723 & 0 & 0 & 0 & 0 & 0 \\ 9.471 & 0.2258 & -4.744 & 0 & 0 & 0 & 0 \\ 0 & 6.874 & -2.207 & -0.8476 & 0 & 0 & 0 \\ 0 & 0 & -1.754 & 3.473 & -0.5604 & 0 & 0 \\ 0 & 0 & 0 & -7.554 & -2.25 & -3.354 & 0 \\ 0 & 0 & 0 & 0 & -0.3107 & 5.124 & -2.304 \\ 0 & 0 & 0 & 0 & 0 & -3.518 & 2.318 \end{bmatrix}$$

```
bidiagonal(n):=matrix(n,n,(i,j)->if(i==j||i==j+1)rand(-10..10)(); else  0; )
```

For example, by writing in Xcas:

bidiagonal(7), the output is:

$$\begin{bmatrix} -9.249 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9.617 & -9.437 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2.179 & 3.265 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3.96 & -8.729 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6.101 & 8.91 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.325 & 3.381 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6.047 & -4.223 \end{bmatrix}$$

In the examples above nonzero entries are random numbers with a 1-digit integer part, generated by Xcas function rand(-10..10).

3.4 *Band Matrices of Special Forms*

Let us now define the n-th order Fibonacci Matrix F_n as a n×n band matrix that has 1's on the main diagonal, -1's along the diagonal immediately above the main diagonal, 1's along the diagonal immediately below the main diagonal and zeros everywhere else. Our programmed function fibonacci(n) generates Fibonacci matrices of size n:

fibonacci(n):=matrix(n,n,(i,j)->if(i==j+1||i==j)1;else (if(i==j-1) -1; else 0;);)

For example, by writing in Xcas:

fibonacci(8), the output is:

$$\begin{bmatrix} 1, & -1, & 0, & 0, & 0, & 0, & 0, & 0 \\ 1, & 1, & -1, & 0, & 0, & 0, & 0, & 0 \\ 0, & 1, & 1, & -1, & 0, & 0, & 0, & 0 \\ 0, & 0, & 1, & 1, & -1, & 0, & 0, & 0 \\ 0, & 0, & 0, & 1, & 1, & -1, & 0, & 0 \\ 0, & 0, & 0, & 0, & 1, & 1, & -1, & 0 \\ 0, & 0, & 0, & 0, & 0, & 1, & 1, & -1 \\ 0, & 0, & 0, & 0, & 0, & 0, & 1, & 1 \end{bmatrix}$$

3.5 *Some Special Matrices*

Defining the n×n matrix $F_n=[f_{ij}]$ for which $f_{ij}=w^{ij}$, i,j=1..n as a Fourier matrix then our programmed function fourier(n) generates Fourier matrices of size n:

fourier(n):=matrix(n,n,(i,j)->w^(i*j))

For example, by writing in Xcas:

fourier(7), the output is:

$$\begin{bmatrix} 1, & 1, & 1, & 1, & 1, & 1, & 1 \\ 1, & w, & w^2, & w^3, & w^4, & w^5, & w^6 \\ 1, & w^2, & w^4, & w^6, & w^8, & w^{10}, & w^{12} \\ 1, & w^3, & w^6, & w^9, & w^{12}, & w^{15}, & w^{18} \\ 1, & w^4, & w^8, & w^{12}, & w^{16}, & w^{20}, & w^{24} \\ 1, & w^5, & w^{10}, & w^{15}, & w^{20}, & w^{25}, & w^{30} \\ 1, & w^6, & w^{12}, & w^{18}, & w^{24}, & w^{30}, & w^{36} \end{bmatrix}$$

3.6 Elementary Matrices

If we define the matrix that subtracts a multiple l of row j from row i as the elementary matrix E_{ij}, with 1's on the diagonal and the number –l in row i, column j then this differs from the identity matrix by one single elementary row operation. Our programmed function elementary(n,k,l,a) takes as arguments matrix size (n), the number of row (k) and column (l) of element a and element (a) and returns the corresponding elementary matrix.

```
elementary(n,k,l,a):= matrix(n,n,(i,j)->if(i==j) 1; else  (if(i==k-1&&j==l-1)a;else 0;);)
```

For example, by writing in Xcas:

elementary(4,3,2,-s), the output is:

$$\begin{bmatrix} 1, & 0, & 0, & 0 \\ 0, & 1, & 0, & 0 \\ 0, & -s, & 1, & 0 \\ 0, & 0, & 0, & 1 \end{bmatrix}$$

3.7 Polynomial Coefficient Matrices

Relying on Schur Theorem (see Chiang, 1984, pp. 601-602) then the real polynomial

$$f(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_n = 0$$

is called Schur stable if its roots x_i are $|x_i| < 1$. The condition $|x_i| < 1$ holds if and only if the n determinants Δ_i (i=1,…,n) are all positive. The determinants Δ_i are:

$$\Delta_1 = \begin{vmatrix} a_0 & a_n \\ a_n & a_0 \end{vmatrix}, \Delta_2 = \begin{vmatrix} a_0 & 0 & a_n & a_{n-1} \\ a_1 & a_0 & 0 & a_n \\ a_n & 0 & a_0 & a_1 \\ a_{n-1} & a_n & 0 & a_0 \end{vmatrix}, \ldots, \Delta_n = \begin{vmatrix} a_0 & 0 & \ldots & 0 & a_n & a_{n-1} & \ldots & a_1 \\ a_1 & a_0 & \ldots & 0 & 0 & a_n & \ldots & a_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n-1} & a_{n-2} & \ldots & a_0 & 0 & 0 & \ldots & a_n \\ a_n & 0 & \ldots & 0 & a_0 & a_1 & \ldots & a_{n-1} \\ a_{n-1} & a_n & \ldots & 0 & 0 & a_0 & \ldots & a_{n-2} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_1 & a_2 & \ldots & a_n & 0 & 0 & \ldots & a_0 \end{vmatrix} \quad (1)$$

Our programmed function schur(poly,var,k) with arguments the polynomial (poly), its variable (var) and the order k (with k ranging from 0 to degree of polynomial minus 1) of the sequence (1), returns the k-th matrix of the Schur theorem. The codes in Xcas are:

```
A11(poly,var):=matrix(degree(poly,var),degree(poly,var),(j,k)->if(j<k) 0 ; else coeff(poly,var)[[j-k+1]];):;
A12(poly,var):=matrix(degree(poly,var),degree(poly,var),(j,k)->if(j>k) 0 ; else coeff(poly,var)[[degree(poly,var)+j-k+1]];):;
A21(poly,var):=matrix(degree(poly,var),degree(poly,var),(j,k)->if(j<k) 0 ; else coeff(poly,var)[[degree(poly,var)+k-j+1]];):;
A22(poly,var):=matrix(degree(poly,var),degree(poly,var),(j,k)->if(j>k) 0 ; else coeff(poly,var)[[k-j+1]];):;
schur(poly,var,k):=blockmatrix(2,2,[subMat(A11(poly,var),0,0,k,k),subMat(A12(poly,var),0,0,k,k),subMat(A21(poly,var),0,0,k,k),subMat(A22(poly,var),0,0,k,k)]):;
```

The Schur theorem is considered as a perfect difference equation counterpart of the Routh theorem in the differential equation setup[3]. Relying now on the Routh-Hurwitz criteria[4] then for the real polynomial

$$f(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ldots + a_n = 0$$

the real parts of all its roots x_i are negative if and only if the n determinants Δ_i are all positive

[3] See among others Brauer and Nohel (1989), Cushing (2004), Moler, Van Loan (1978), Moler ,Van Loan (2003) and Noble (1969).

[4] For more details on the theorem see among others Samuelson (1947, pp. 429-435).

$$\Delta_1 = |a_1|, \Delta_2 = \begin{vmatrix} a_1 & a_0 \\ a_3 & a_2 \end{vmatrix}, \Delta_3 = \begin{vmatrix} a_1 & a_0 & 0 \\ a_3 & a_2 & a_1 \\ a_5 & a_4 & a_3 \end{vmatrix}, \ldots, \Delta_n = \begin{vmatrix} a_1 & a_0 & 0 & 0 & \ldots & 0 \\ a_3 & a_2 & a_1 & a_0 & \ldots & 0 \\ a_5 & a_4 & a_3 & a_2 & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & a_n & a_{n-1} \\ 0 & 0 & 0 & 0 & \ldots & a_n \end{vmatrix} \quad (2)$$

Our programmed function routh(poly,var,t) with arguments the polynomial (poly), its variable (var) and the order t (with t ranging from 0 to degree of polynomial minus 1) of the sequence (2), returns the t-th matrix of Routh's theorem. The codes in Xcas are:

```
routh(poly,var,t):=subMat(blockmatrix(degree(poly,var),1,[seq(list2mat(
[seq(if(j<=degree(poly,var))coeff(poly,var)[[j+1]];else  0;,j=k..0)],2*degree(poly,var))
,k=1..2*degree(poly,var),2)]),0,0,t,t):;
```

Applications with Schur's theorem and Routhian analysis in Economic problems can be found in Halkos and Tsilika (2012a).

The matrix with entries the coefficients of the variables (both endogenous and predetermined) excluded from an equation of a simultaneous equation model but included in the other equations of the model, has a role to play in rank condition of identifiability. Coefficient matrices related to rank condition of identifiability are generated by programmed functions in Xcas in Halkos and Tsilika (2012b).

3.7.1 Numerical Examples

Let us see next some numerical examples of the two theorems mentioned so far.

By writing in Xcas:

schur(a0*x^4+a1*x^3+a2*x^2+a3*x+a4,x,3), the output is:

$$\begin{bmatrix} a0, & 0, & 0, & 0, & a4, & a3, & a2, & a1 \\ a1, & a0, & 0, & 0, & 0, & a4, & a3, & a2 \\ a2, & a1, & a0, & 0, & 0, & 0, & a4, & a3 \\ a3, & a2, & a1, & a0, & 0, & 0, & 0, & a4 \\ a4, & 0, & 0, & 0, & a0, & a1, & a2, & a3 \\ a3, & a4, & 0, & 0, & 0, & a0, & a1, & a2 \\ a2, & a3, & a4, & 0, & 0, & 0, & a0, & a1 \\ a1, & a2, & a3, & a4, & 0, & 0, & 0, & a0 \end{bmatrix}$$

routh(a0*x^6+a1*x^5+a2*x^4+a3*x^3+a4*x^2+a5*x+a6,x,5), the output is:

$$\begin{bmatrix} a1, & a0, & 0, & 0, & 0, & 0 \\ a3, & a2, & a1, & a0, & 0, & 0 \\ a5, & a4, & a3, & a2, & a1, & a0 \\ 0, & a6, & a5, & a4, & a3, & a2 \\ 0, & 0, & 0, & a6, & a5, & a4 \\ 0, & 0, & 0, & 0, & 0, & a6 \end{bmatrix}$$

3.8 Functional Matrices

Let us now define the matrix of the first order partials of a function as the Jacobian matrix. Our programmed function jacobian(listf,vars) takes as arguments the list of functions (listf) and the variable vector (vars) and returns the jacobian matrix:

jacobian(listf,vars):=transpose(diff(listf,vars))

The bordered Hessian matrix of a function $f(x_1,...,x_n)$ subject to m constraints (m<n) of the form $g^j(x_1,...,x_n)$ appears as

$$B = \begin{pmatrix} 0 & \dots & 0 & g_1^1 & \dots & g_n^1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & g_1^m & \dots & g_n^m \\ g_1^1 & \dots & g_1^m & f_{11} & \dots & f_{1n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ g_n^1 & \dots & g_n^m & f_{n1} & \dots & f_{nn} \end{pmatrix}$$

Our programmed function borderhessian(f,vars,listconst,#const) returns the bordered hessian matrix of a function subject to m equality constraints. borderhessian function takes as arguments the function (f), the variable vector (vars), a vector containing the constraints' formulas (listconst) and the number of the constraints (#const). borderhessian function in Xcas is well defined by the following codes, given that the jacobian function has earlier been defined.

borderhessian(f,vars,listconst,#const):=blockmatrix(2,2,[newMat(#const,#const),jacobian(listconst,vars),transpose(jacobian(listconst,vars)),hessian(f,vars)])

Let us suppose that *y1(x), y2(x),…,yn(x)* are (n-1) times differentiable functions. Then the Wronskian of these functions is defined as the matrix

$$W(y_1, y_2, \ldots, y_n) = \begin{pmatrix} y_1' & y_1' & \cdots & y_1' \\ y_1 & y_2 & \cdots & y_n \\ \cdots & \cdots & \cdots & \cdots \\ y_1^{(n-1)} & y_2^{(n-1)} & \cdots & y_1^{(n-1)} \end{pmatrix}$$

Our programmed function wronskian(listf,var) returns the wronskian matrix of a set of functions (listf) of variable (var):

wronskian(listf,var):=seq(diff(listf,var$n),n,0,length(listf)-1)

3.8.1 Numerical Examples

Let us see next some numerical examples of the functional matrices mentioned so far. Specifically we have

jacobian([x^3*y,x^2*y^2],[x,y])

$$\begin{bmatrix} 3\cdot x^2\cdot y, & x^3 \\ 2\cdot x\cdot y^2, & x^2\cdot 2\cdot y \end{bmatrix}$$

borderhessian(x^2+y^2+w^2,[x,y,w],[x+2*y+3*w,2*x+3*y+w-4],2)

$$\begin{bmatrix} 0, & 0, & 1, & 2, & 3 \\ 0, & 0, & 2, & 3, & 1 \\ 1, & 2, & 2, & 0, & 0 \\ 2, & 3, & 0, & 2, & 0 \\ 3, & 1, & 0, & 0, & 2 \end{bmatrix}$$

wronskian([x^3+3,sqrt(x^2+1),x*sin(x)],x)

$$\begin{bmatrix} x^3+3, & \sqrt{x^2+1}, & x\cdot\sin(x) \\ 3\cdot x^2, & \dfrac{x\cdot\sqrt{x^2+1}}{x^2+1}, & x\cdot\cos(x)+\sin(x) \\ 6\cdot x, & \dfrac{\sqrt{x^2+1}}{x^4+2\cdot x^2+1}, & (-x)\cdot\sin(x)+2\cdot\cos(x) \end{bmatrix}$$

4. Conclusions

Working in Xcas environment, a user has the option to use Xcas' built-in functions for matrix operations and manipulation. Xcas is free of any charges accessible to all users interested. Programming structure in Xcas is simple and programs can be inserted in the same session with entries of different types (symbolic, numerical, graphical computations). In addition, our codes suggest a direction for computer experiments. They constitute an open source for further calculations and give ideas for efficient computation.

Our matrix generator has many advantages.

- The programmed matrix functions are not included in typical / commonly used algebra packages and produce output requiring simple and clear input.
- Some of our functions have a code structure which uses random numbers to produce random block diagonal matrices, random triangular matrices, random band matrices, offering infinite number of examples.
- In educational practice and in research, by automatic construction of the matrix needed, the user avoids the problem of input and saves time.
- In case of coefficient matrices, the user avoids complex laws of formation and consequently, possible mistakes.
- In case of functional matrices the user also avoids differential calculus operations.

References

Anton H. (2000). *Elementary Linear Algebra.* John Wiley & Sons, Inc., New York.

Anton H., Busby R.C., Knoll C. and Martinez-Garza C. (2003). *Contemporary Linear Algebra*, J. Willey, Hoboken NJ.

Batten, D and Martellato, D. (1985). Classical versus Modern Approaches to Interregional Input-Output Analysis. *The Annals of Regional Science*, **19(3):** 1-15.

Brauer F., Nohel J.A. (1989). *Qualitative Theory of Ordinary Differential Equations*. Reprint, Dover.

Cassetti M. (1995). A new method for the identification of patterns in input-output matrices. *Economic Systems Research*, **7**: 363-381.

Chiang A. (1984). *Fundamental Methods of Mathematical Economics.* 3rd edition, McGraw-Hill Book, Singapore.

Cushing J.M. (2004). *Differential Equations: An Applied Approach*. Prentice Hall.

Elsner L. and Johnson C.R. (1989). Nonnegative matrices, zero patterns and spectral inequalities. *Linear Algebra Applications*, **120**: 225-236.

Goldberg J. L. (1991). *Matrix theory with applications. McGraw-Hill, Inc.*, Columbus.

Halkos G.E. and Tsilika K.D. (2012a). Stability Analysis in Economic Dynamics: A Computational Approach. MPRA paper 41371, University Library of Munich, Germany.

Halkos G.E. and Tsilika K.D. (2012b). Programming identication criteria in simultaneous equation models. MPRA paper, 43467, University Library of Munich, Germany.

Hall F.J., Li Z. and Wang D. (2001). Symmetric sign pattern matrices that require unique inertia. *Linear Algebra Applications*, **338(1-3)**: 153-169.

Heppell S.S., Caswell H. and Crowder L.B. (2000). Life history and elasticity patterns, perturbation analysis for species with minimal demographic data. *Ecology,* **81(3)**: 654-665.

Hertz G.Z., Hartzell G.W. III and Stormo G.D. (1990). Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Computer Applications in the Biosciences* **6(2):** 81-92.

Johnson C.R. (1983). Sign patterns of inverse nonnegative matrices. *Linear Algebra Applications,* **55**: 69-80.

Kaven A. and Sayarinejad M.A. (2004). Eigensolutions for factorable matrices of special patterns. *Communications in Numerical Methods in Engineering*, **20**: 133-146.

Lipschutz S. (1987). *Theory and problems of linear algebra. McGraw-Hill, Inc*., Columbus.

Lloyd W.P. and Lee C.F. (1976). Block Recursive Systems in Asset Pricing Models, *The Journal of Finance*, **31(4):** 1101-1113.

McDonald J.J., Olesky D.D., Tsatsomeros M.J. and van den Driessche P. (1997). Ray patterns of matrices and nonsingularity. *Linear Algebra Applications,* **267**: 359-373.

Moler C., Van Loan C. (1978). Nineteen dubious ways to compute the exponential of a matrix. *SIAM Review*, **20**: 801-836.

Moler C., Van Loan C. (2003). Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, **45**: 3-49.

Noble B. (1969). *Applied Linear Algebra*. Prentice Hall.

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

Quarteroni A. and Saleri F. (2006). *Scientific Computing with MATLAB and Octave*, 2nd edition, Springer-Verlag, Heidelberg, 2006.

Samuelson P.A. (1947). *Foundations of Economic Analysis*. Harvard Univrersity Press.

Stadelmaier M.W., Rose N.J., Poole G.D. and Meyer C.D. (1982). Nonnegative matrices with power invariant zero patterns. Linear Algebra Applications, **42:** 23-29.

Strang G. (1988). *Linear Algebra and its Applications*. 3rd edition, Harcount Brace Jovanovich College, Philadelphia, New York.

Tardos G. (2005). On 0-1 matrices and small excluded submatrices, *Journal of Combinatorial Theory Series A,* **111(2):** 266-288.

Tarr D.G. (1976). Distributed Lags, Morishima Matrices, and the Stability of Economic Models. *Econometrica*, **44(3)**: 597-600.

Veinott A.F. (1969). Minimum concave-cost solution of Leontief substitution models of multi-facility inventory systems, Operational Research, **17(2)**: 262-291.

Wermuth N. (1992). On block-recursive linear regression equations. *Revista Brasileira de Probabilidade e Estatistica,* **6**, 1-56.

Wold H. (1954). Causality and Econometrics. *Econometrica*, **22**: 162-177.