Implementing a Generalized Cobb Model for Production Functions

Alina Andreica
IT Department
Babes-Bolyai University, Cluj-Napoca, Romania
alina.andreica@ubbcluj.ro

Florina Covaci
IT Department
Babes-Bolyai University, Cluj-Napoca, Romania
florina.covaci@ubbcluj.ro

***Abstract* — The paper proposes an extension of Cobb-Douglas model for production functions applicable in multi-product and regional contexts and a combined symbolic & numeric method for solving the system. In this respect, we propose a new application of Buchberger's algorithm for computing Gröbner basis in simplifying the polynomial set that is obtained by modelling economic production systems. We present our Mathematica categorical implementation of Gröbner basis algorithm that can be applied for performing necessary computations. We apply Gröbner basis algorithm for some specific production function cases proposed in the economic literature and discuss the results. We consider that Gröbner basis algorithm is important in processing (according to various variable orderings) the polynomial set that is constructed and in synthesizing the most important economic characteristics taken into account by the model. Further on, numeric methods can be applied if necessary. We also note the importance of generic implementations of Buchberger's algorithm, which can be easy adapted for various domains.**

Keywords-Cobb-Douglas model; generalized production functions; symbolic modelling; Buchberger algorithm; generic implementations.

I. INTRODUCTION AND WORKING FRAMEWORK

Category theory in symbolic computation introduces techniques for implementing general working contexts of performing symbolic algorithms. The defined categories can later be particularized for various domains, by elegantly using the same definition. Definitions for categories and domains, from the symbolic point of view, can be found in [1], [2]. In these papers, we describe an extension of Mathematica – a computer algebra or symbolic computation system [15] – with a new type system, containing algebraic categories and abstract definitions of Gröbner basis [4] simplification algorithm. Within this package, the algorithms that implement Buchberger's method for computing the Gröbner basis and the reduced Gröbner basis for a given set of polynomials are applied for multivariate polynomial domains over various coefficient rings.

Buchberger's algorithm for Gröbner basis [4] has found numerous applications in various fields related to polynomial simplification over various fields [10], [11] including computational geometry [12].

Within this paper, we present a new application of Buchberger's algorithm for Gröbner basis, belonging to the economic field, namely, for simplifying production function sets according to Cobb-Douglas model [7].

Section 2 describes the Cobb-Douglas model for economic production functions [7]. Section 3 presents the basic principles of Buchberger algorithm for computing the Gröbner basis. In Section 5, we present the model we propose for generalizing the Cobb Douglas model at a macroeconomic scale, taking into account many countries. In Section 6, we describe the basic principles of categorical polynomial definition and processing from our Mathematica implementation, while Section 7 presents our implementation of computing the Gröbner basis. Conclusions reveal the most important contributions of the paper.

II. COBB-DOUGLAS MODEL FOR ECONOMIC PRODUCTION FUNCTIONS

In economic modelling, the Cobb-Douglas functional form of production functions [7] is widely used to represent the relationship of a production output in respect with specific inputs. The model was proposed by Knut Wicksell [14], and tested against statistical evidence by Paul Douglas and Charles Cobb in 1928 [7].

The model states [7] that a production function can be written in the functional form:

$$Y = AL^{\alpha}K^{\beta} \tag{1}$$

where:

Y the production output

L represents the labour input

K represents the capital input

A, α and β are constants determined by technology. The exponents α and β are output elasticity coefficients with respect to labour and capital, respectively. Output elasticity measures the responsiveness of output to a change in levels of either labour or capital used in production [7].

According to the original model [7], the following cases are considered relevant:

- if $\alpha+\beta=1$, the production function has constant returns to scale;
- if $\alpha+\beta<1$, returns to scale are decreasing;
- if $\alpha+\beta>1$ returns to scale are increasing. Assuming perfect competition, α and β can be shown to be labour and capital's share of output.

Cobb and Douglas were influenced by statistical evidence that appeared to show that labour and capital shares of total output were constant over time in developed

 ISBN: 978-1-61208-290-5

countries; they explained this feature by applying statistical fitting in least-squares regression of their production function [7].

III. BUCHBERGER'S ALGORITHM

Buchberger's algorithm computes the Gröbner basis [4] of a given set of polynomials over a coefficient ring as a "simplified" polynomial set; the Gröbner basis G for a polynomial set F is a "reduced" form of G [4], [5] that generates the same ideal as F.

The "simplified" polynomial set that is obtained in the reduction process is significant in solving various problems involving polynomial sets. Buchberger synthesizes [5] a variety of application fields for Gröbner basis algorithm, revealing its importance in systems theory. The application presented in this paper is founded on the principles presented in [5].

Based on category theory and generic principles, recent versions of Buchberger's algorithm [6] are abstract implementations, which offer generic frameworks for applying the algorithm. The implementation we presented for Mathematica in [1] and [3] are based on these generic principles, proving to have important flexibility and extendibility advantages in applying the algorithm over various domains

Present built-in Mathematica implementations of Buchberger algorithm support polynomials over the following coefficient domains: InexactNumbers, Rationals, RationalFunctions and Polynomials[*x*] [16].

IV. GENERALIZED COBB-DOUGLAS MODEL. APPLYING BUCHBERGER'S ALGORITHM

We generalize the Cobb-Douglas model for production functions in order to represent production states at a macroeconomic scale. We consider that such a model is relevant in order to better grasp the production phenomena that appear in economies.

In this respect, we propose a set of Cobb-Douglas functions for expressing the necessary labour input and capital input to obtain a certain output within an economy. Taking into consideration the economy of a country, we consider the variables Lp, Kp as the labour input, respectively capital input for obtaining the product p and Gross Domestic Product (GDP) as the global output. For more economic regions, Gross Domestic Products (GDPs) can be represented as:

$Y_i, \mathrm{i} = 1, \mathrm{m}$, - $\mathrm{Y_i}$ being the Gross Domestic Product (GDP) generated by region i. If we consider that each region i produces n_i products, we arrive to the generalized model (2).

We consider that the model we propose is relevant for expressing production characteristics at a macroeconomic scale. Furthermore, the possibility of expressing production characteristics in regional development frameworks, can be very useful in the context of economic European integration.

$$\begin{cases} Y_1 = A_{11}L_{p_1}^{\alpha_{11}}K_{p_1}^{\beta_{11}} + A_{12}L_{p_2}^{\alpha_{12}}K_{p_2}^{\beta_{12}} + \ldots + A_{1n_1}L_{p_{n_1}}^{\alpha_{1n_1}}K_{p_{n_1}}^{\beta_{1n_1}} \\ Y_2 = A_{21}L_{p_1}^{\alpha_{21}}K_{p_1}^{\beta_{21}} + A_{22}L_{p_2}^{\alpha_{22}}K_{p_2}^{\beta_{22}} + \ldots + A_{2n_2}L_{p_{n_2}}^{\alpha_{2n_2}}K_{p_{n_2}}^{\beta_{1n_2}} \\ \quad\vdots \\ Y_m = A_{m1}L_{p_1}^{\alpha_{m1}}K_{p_1}^{\beta_{m1}} + A_{m2}L_{p_2}^{\alpha_{m2}}K_{p_2}^{\beta_{m2}} + \ldots + A_{mn_m}L_{p_{n_m}}^{\alpha_{mn_p}}K_{p_{n_m}}^{\beta_{mn_m}} \end{cases} \quad (2)$$

Consequent to defining the functional production polynomials, an important task is to simplify the polynomial set in order to obtain in a canonical form the functions that express labour and capital inputs for different products and regions at a macroeconomic scale. Such a form is important for characterizing the production features in a synthesized manner. In order to obtain this simplified form, we can apply Buchberger's algorithm.

For monomial exponents representing elasticity coefficients that have real values, we can apply the reduction steps by computing the necessary subtractions of the exponent lists: in respect with the order relation on the real set $\Re$, extended to tuples of real exponents, $\Re_n$, the "leading" monomial of each polynomial can be reduced by computing monomials with corresponding exponent subtractions in respect with the other polynomials. For the reduction step in Buchberger's algorithm, instead of taking into account the least common multiple for pairs of exponent vector lists, we compute the maximum of two exponent vector lists in respect with $\Re_n$

In our Mathematica implementation based on generic principles [1], [3], we easily defined the Lcm (Least Common Multiple) operator in the exponent vector package *VectExp* as a Max operator and we supplementary defined the real domain in our coefficient domain package *DomCoef* – for which a code overview is given in Fig. 1:

```
DomReal[R_]:=Module[{},
  InelCom[R,"+","*","0","1"];
  (* Mathematica definition [2] *)
  R["+",a_Real,d_Real]:=a+d;
  R["+",a_Real,Infinity]:=Infinity;
  R["+", Infinity, d_Real]:=Infinity;
  R["-",a_Real,b_Real]:=a-b;
  R["*",a_Real,b_Real]:=a*b;
  R["/",a_Real,b_Real]:=a/b;
  R["=",a__,b__]:=SameQ[a,b];
  R["<>",a__,b__]:=UnsameQ[a,b];
  R["<",a__,b__]:=a<b;
  R["<=",a__,b__]:=a<=b;
  R[">",a__,b__]:=a>b;
  R[">=",a__,b__]:=a>=b;
  R["0"]:=0;
  R["max"]:=Infinity;
  R["/",a_List,b_Real]:=a/b;
  R["+",a__,b__]:=a+b;
  R["-",a__,b__]:=a-b;
  R["*",a__,b__]:=a*b;
  R["/",a_List,b_Real]:=a/b;
  R["/",a__,b__]:=a/b;  ]
```

Figure 1. Mathematica abstract definition for the real domain.

The higher impact of labour L or capital K variables in the above described polynomials – (2) – can be evaluated by taking into account appropriate orderings of L_i, i=1,m, K_i, i=1,m. Moreover, the model can be enhanced by supplementary taking into account new production variables. In this respect, [17] proposes to take into account a third variable as a sum of variables with a smaller production impact. Still, the model we propose may take into account any number of variables.

We applied Buchberger's algorithm for sets of 2-3 polynomials representing GDPs in 2 countries by taking input values and elasticity coefficients proposed in [17].

For cases in which the reduced polynomial set should be further processed, numeric solving methods may be consequently applied

V. POLYNOMIAL CATEGORICAL IMPLEMENTATION

In order to implementing the multivariate polynomial category, we define an auxiliary domain for monomials [1], naming it the exponent vector domain [9].

An exponent vector with a given base of identifiers (for example, x,y,z) will be retained by the list of exponents corresponding to each variable. In operating upon exponent vectors, we use two types of representations: lists, respectively products of primes with the exponents in question [9]. We use the latter representation [1], the exponent vector e_1, e_2, ..., e_n will be retained and processed by the number

$$p_1^{e_1} * p_2^{e_2} * ... * p_n^{e_n}, \quad (3)$$

where p_1, p_2, ..., p_n are prime numbers, for simplifying computations - the first prime numbers. In this representation, an exponent vector sum reduces to the corresponding prime numbers product, whereas the greatest common divisor (Gcd) and the least common multiple (Lcm) can be computed by similar operations upon prime products.

The exponent vector domain is an abelian monoid [2] that introduces the following operations:

- computing neutral, minimum and maximum elements ("0", "Max","Min");
- conversions between the list and number internal forms ("ListaInVectorNr", "VectorNrInLista");
- sum ("+"), greatest common divisor ("Gcd") and least common multiple ("Lcm") for two exponent vectors;
- positiveness test for an exponent vector ("Pozitiv");
- divisibility test for two exponent vectors ("|");
- relational operators ("<", ">", "=", "<=", ">=", "<>") between two exponent vectors - we shall consider the lexicographical ordering corresponding relations are implemented by string[...] functions);
- conversions between external and internal forms ("Inp", "Out").

We give in Fig. 2 an overview of the Mathematica package which defines the exponent vector domain (for the functions in italics we omitted the body). All operations within an exponent vector domain V, created by the function VectExp[V,lv], where lv is the variable list representing the base, will be prefixed with the domain name and will have as the first parameter the operation code. For example, V["+",v1,v2] returns the sum of two exponent vectors (in internal form), V["Gcd",v1,v2] computes the greatest common divisor, while V["Lcm",v1,v2] computes the least common multiple.

```
BeginPackage["VectoriExp‘"]
VectExp::usage="VectExp[V ,lv List] defines the
exponent vector domain V, with the base lv "
Intreg::usage="Integer domain"
string::usage="string operations"
Begin["VectoriExp‘Private‘"]
Needs["HierMath‘"]
string["Rel",s1 String,s2 String]:= Mathematica definition[2]
(*tests the relation <, =, > between s1 and s2, returning -1,0,1*)
(* "Rel" may be "<", ">", "=", "<>" *)
VectExp[V ,lv List]:= Mathematica definition [2] (*creates the
exponent vector domain V, with the base lv*)
MonoidCom[V,"+",en]; Mathematica definition [2]
(*creates an abelian monoid [2]*)
V["ListaInVectorNr",l List]:= Mathematica definition [2]
V["VectorNrInLista",nr Integer]:= Mathematica definition[2]
V["+",l1 List,l2 List]:=l1+l2;
V["-",l1 List,l2 List]:=l1-l2;
V["Pozitiv",l List]:= Mathematica definition [2]
(*tests if all list elements are >0*)
V["Gcd", l1 List, l2 List]:= Mathematica definition [2]
(*computes greatest common divisor *)
V["Lcm", l1 List, l2 List]:= Mathematica definition [2]
(computes teast common multiple *)
V["Rel", l1 List, l2 List]:= Mathematica definition [2] (*tests the
relation <, =, > between l1 and l2, returning -1,0,1*)
V["|",l1 List,l2 List]:= Mathematica definition[2]
(* divisibility test *)
V["Out",l List]:= Mathematica definition [2]
(* output form *)
V["Inp",e ]:= Mathematica definition [2] (*transforms an input with
the syntax x[^e1]*y[^e2]... into the internal list form;
the code is rather complex and based on Mathematica
internal forms*)
V["Max"]:=max; V["Min"]:=min;]
End[]
EndPackage[]
```

Figure 2. Mathematica abstract definition for the exponent vector domain.

The representation of a polynomial (polinom.m package) uses a list of two elements: the exponent vector list, lexicographically ordered, and the corresponding coefficient list [2]. For example, the polynomial 2*xˆ2*z-5*y (with the base {x,y,z}) will be represented as {{{0,1,0},{2,0,1}}, {-5,2}}

We give below the main part of the Mathematica package which defines the polynomial category (polinom.m package [2]; we omitted the bodies for the functions in italics). Within Mathematica definitions, functional and parametric specification of operations within various domains can be noticed (Figs. 1, 2).

 ISBN: 978-1-61208-290-5

For example, within the polynomial domain Pol, defined by Polinom[Pol, DCoef, DVect, l], where DCoef is the coefficient domain and DVect is the exponent vector domain with the base l, the construction DVect["|",v,Pol["Monom",i,P]] performs a divisibility test - within DVect exponent vector domain - between two exponent vectors, the second one corresponding to a monomial selected from a polynomial P (by "Monom" operation, within the polynomial domain Pol). DVect["+",o1,o2] is the sum of o1 and o2 in DVect domain, whereas DCoef["+",o1,o2] is a sum in DCoef domain. An overview of the Gröbner basis package we have defined [1], [2] is given in Fig. 3:

```
BeginPackage["Polinom`"]
Polinom::usage="Polinom[Pol,DCoef,DVect,l] defines the
multivariate polynomial domain Pol, over the coefficient domain
DCoef and the exponent vector domain DVect (of monomials), with
the basis l"
Begin["Polinom`Private`"]
Needs["VectoriExp`","DomCoef`"]
Polinom[Pol ,DCoef ,DVect ,baza List]:=Module[n,lv,
(*defines a multivariate polynomial domain, with coefficients in
DCoef, over the exponent vector domain Dvect, with the base
baza*)
InelCom[DCoef,"+","*"]; (*creates an abelian ring [2]*)
VectExp[DVect,baza];
Pol["DomCoef"]:=DCoef;
Pol["DomVectExp"]:=VectExp;
Pol["Init"]:=List[List[DVect["Max"]],List[]];
(*returns the null polynomial, with a sentinel in the exponent vector
list*)
Pol["Nul",P ]:=Mathematica definition [2] ;
(* returns True if P is null *)
Pol["Nr",P ]:=Length[P[[1]]]-1; (*dimension*)
Pol["Monom", i ,P ]:=P[[1]][[i]];
(*the ith monomial from the polynomial P*)
Pol["Coef",i ,P ]:=If[i<=Pol["Nr",P],P[[2]][[i]],0];
Pol["Indice",l ,P ]:= Mathematica definition [2]
(*the index of the monomial l in the polynomial P*)
Pol["MonomGrMax",P ]:=P[[1]][[Pol["Nr",P]]];
Pol["CoefGrMax",P ]:=P[[2]][[Pol["Nr",P]]];
Pol["AdaugMonom",T ,v ,c ]:= Mathematica definition [2]
 (*adds to the polynomial T the monomial formed by the exponent
vector v and the coefficient c taking into account the lexicographical
ordering; if v exponent vector exists, it adds the coefficient c to the
appropriate existing one*)
Pol["MonomInPol",l :List,c :Number]:= Mathematica definition [2]
(*transforms a monomial into the equivalent polyniomial *)
Pol["+",P1 ,P2 ]:= Mathematica definition [2]
(* returns the sum of P1, P2 *)
Pol["*",P1 :List,P2 :List]:= Mathematica definition [2]
(* returns the product of P1, P2 *)
Pol["&",nr :Number,P :List]:= Mathematica definition [2]
(*multiplies P by the number n*)
Pol["-",P1 ,P2 ]:=Module[{P}, (*polynomial subtraction *)
P=Pol["&",-1,P2]; Return[Pol["+",P1,P]]; ];
Pol["/",P ,v List,c :Number]:= Mathematica definition [2]
(* divides each of P's monomials by the exponent vector v and
coefficient c and returns the result *)
Pol["|",v List,P ]:= Mathematica definition [2]
(* tests whether the exponent vector v divides any of P's monomials
and returns True or False *)
Pol["Out",P ]:= Mathematica definition [2] (*polynomial display*)
Pol["Inp",e ]:= Mathematica definition [2] (*transforms an input
polynomial into the internal form; the code is rather complex and
based on Mathematica internal forms*)
] (*Module*)
End[]
EndPackage[]
```

Figure 3. Mathematica abstract definition for the polynomial domain.

The following section is dedicated to the Gröbner basis algorithm and its application to production functions.

VI. GROEBNER BASIS ABSTRACT IMPLEMENTATION AND APPLICATION CASES FOR PRODUCTION FUNCTIONS

We implemented Buchberger's algorithms for computing the Gröbner basis and the reduced Gröbner basis [4] of a polynomial set into Mathematica packages: groebner.m and groebred.m [1], [2]. The functions which compute the Gröbner bases are parameterized with a polynomial domain, therefore they can be applied for polynomial domains over any consistent coefficient domain that is previously defined. Note that a polynomial domain is created by using the polynomial categorical definition within polinom.m package, which is parameterized with a coefficient domain defined in domcoef.m package [1], [2]. Within groebner.m package [1] we implemented Buchberger's Gröbner basis algorithm [4] - BazaGroebner[] function. We completely described the algorithmic iterations for computing the normal form of a polynomial modulo a polynomial set - Normal[Pol,F,g] function, where Pol is the current polynomial domain. For computing the S-polynomial of two polynomials, we implemented the formula proposed in [9] - SPol[] function.

An overview of the Gröbner basis package we have defined [1], [2] is given in Fig. 4:

```
BeginPackage["Groebner`"]
Normal::usage="Normal[Pol,F,g] verifies if g is in normal form mod
F, over the polnomial domain Pol"
FormaNormala::usage="FormaNormala[Pol,DCOef,DVect,F,p]
returns the p's normal form modulo F; operations are
performed over the polynomial domain Pol"
SPol::usage="SPol[Pol,DCoef,DVect,P1,P2] computes
Rez=SPol(P1,P2), in the polynomial domain
Pol(DCoef,DVect)"
BazaGroebner::usage="BazaGroebner[Pol,DCoef,DVect,F] returns,
for F set of polynomials over the domain
Pol(DCoef,DVect), F's Groebner base"
MultPolExtInInt::usage="MultPolExtInInt[Pol,M] transforms a set of
polynomials in external representation into internal representation
(operations over Pol domain)"
MultPolIntInExt::usage="MultPolIntInExt[Pol,M] transforms a set of
polynomials in internal representation into external representation
(operations over Pol domain)"
Tiparire::usage="prints a set of polynomials given in internal
representation"
TipPerechi::usage="prints a set of polynomial pairs given in internal
representation"
Begin["Groebner`Private`"]
Needs["Polinom`"]
PolNormal[Pol ,F List,g ]:= Mathematica definition [2]
(*Verifies whether g is in normal form mod F, i. e. no
```

```
monomial of g is divisible by the leading monomial of
any polynomial belonging to F - set of polynomials.
Operations are performed within the polynomial domain Pol*)
FormaNormala[Pol ,DCoef ,DVect ,F List,p ]:= ]:= Mathematica
definition [2]
(*Returns p's normal form mod F; operations are performed
within the polynomial domain Pol(DCoef,DVect) *)
SPol[Pol ,DCoef ,DVect ,P1 ,P2 ]:= Mathematica definition [2]
(*computes, within the polynomial domain P, Rez=SPol(P1,P2)*)
MultPolIntInExt[Pol ,M List]:= ]:= Mathematica definition [2]
(*transforms M polynomial set from internal form into a set of
external forms*)
MultPolExtInInt[Pol ,M List]:= Mathematica definition [2]
(*transforms M polynomial set from external form into a set of
internal forms *)
Tiparire[Pol ,M List]:= Mathematica definition [2] (*displays a
polynomial set*)
TipPerechi[Pol ,M List]:= Mathematica definition [2]
(*displays a set of polynomial pairs*)
BazaGroebner[Pol ,DCoef ,DVect ,baza List,M List]:= Mathematica
definition [2]
(* returns the Groebner basis of the polynomial set M
(given as a list), within the polynomial domain Pol *)
End[]
EndPackage[]
```

Figure 4. Mathematica abstract definition for the Gröbner basis algorithm.

Groebred.m package [1], [2] implements Buchberger's algorithm for computing the reduced Gröbner basis [4].

Comparing our implementation to the Mathematica built in one GroebnerBasis[{$poly_1$, $poly_2$, …},{x_1, x_2, …}], our abstract based one is obviously slower, but it enables computations over various abstract coefficient domains, which can be defined. The built in GroebnerBasis implementation works over rational numbers, integers, rational functions or inexact numbers, with additional computation options [16].

We have applied the Gröbner basis algorithm in Mathematica for various cases of production functions proposed for Romania and Moldova for the period 2002-2004 in [17]. We have denoted with R the polynomial corresponding to Romania's Gross Domestic Product (GDP) function for this period and with M – the polynomial corresponding to Moldova's Gross Domestic Product (GDP) function for the same period.

For the set of polynomial production functions:

$R = 37.4^a\ 62.6^b\ X+45.3^a\ 54.7^b\ Y+49.1^a\ 50.9^b\ Z$

$M = 33.8^a\ 66.2^b\ X+33.4^a\ 66.6^b\ Y+40.5^a\ 59.5^b\ Z$

where X, Y, Z are technology based variables, and using symbolic elasticity coefficients a, b [17], the Gröbner basis still contains two polynomials in X, Y, Z:

$\{-1531.14^a\ 3621.14^b\ Y+1249.16^a\ 4169.16^b\ Y-1659.58^a\ 3369.58^b\ Z+1514.7^a\ 3724.7^b\ Z,33.8^a\ 66.2^b\ X+33.4^a\ 66.6^b\ Y+40.5^a\ 59.5^b\ Z,37.4^a\ 62.6^b\ X+45.3^a\ 54.7^b\ Y+49.1^a\ 50.9^b\ Z\}$

Using the α, β elasticity proposed in [17] as a, b values, we obtain the liniar polynomials:

$R = 45.3457\ X+48.6099\ Y+49.7656\ Z$

$M = 43.4613\ X+43.2359\ Y+46.7666\ Z$

The GroebnerBasis function reduces the X variable (corresponding to year 2002) from the R polynomial and the Y variable (corresponding to year 2003) from the M polynomial, generating the following simplified set:

$\{1.\ Y+0.27755\ Z,1.\ X+0.799942\ Z\}$

We may infer that during the period 2002-2004, for Romania and Moldova, the most relevant evolution years from the production point of view were 2003 and 2004 for Romania and 2002 and 2004 for Moldova. Such results have to be correlated with other macroeconomic variables.

Another possible application case would be the one taking into account countries from Latin America, considering the K values as capital flows and the L values given in [18]. Simplified polynomials would mean similar evolutions in different countries.

VII. CONCLUSION AND FUTURE WORK

We addressed a problem from an economic field, namely a generalized model for production functions, by applying computer algebra tools.

We generalized Cobb-Douglas model for production functions in multi-product and regional contexts by constructing a representative polynomial set in respect with the production inputs (labour, capital, other variables) and we propose the application of Buchberger's algorithm in simplifying the polynomial set that is obtained. We consider that Gröbner basis algorithm is important in processing (according to various variable orderings) the polynomial set that is constructed and in synthesizing the most important economic characteristics taken into account by the model.

We presented our Mathematica categorical implementation of Buchberger's algorithm for Gröbner basis algorithm that can be applied for performing necessary computations. We underline the importance of such abstract implementations, which can be easy adapted for various domains based on parameterized principles. Our implementation is actually an extension of Mathematica with a type system.

We applied the Gröbner basis algorithm for Gross Domestic Product (GDP) functions of Romania and Moldova for the period 2002-2004 using data proposed by Zaman et al in [17] and we discuss the results of applying Buchberger's simplification algorithm on these polynomial sets. Similar processings can be performed for other countries, using specific values that are available in economic analyses for the input data in the production functions.

We intend to further work on the proposed model and to study other cases from the economic literature. We also intend to extend our implementation of Buchberger algorithm for new domains, based on the same abstract principles.

REFERENCES

[1] A. B. Andreica, "Parameterized Types for Categorical Definitions in Mathematica", Symbolic and Numeric Algorithms for Scientific Computing - SYNASC 2002 International Workshop, Mirton, Editors: D. Petcu, V. Negru, D. Zaharie, T. Jebelean, 2002, pp. 8-25.

[2] A. B. Andreica, "Defining Algebraic Categories in Mathematica", Analele Universitatii de Vest Timisoara, Seria Matematica - Informatica, Categ CNCSIS B+, XLI, 2003, pp. 9 - 23

 ISBN: 978-1-61208-290-5

[3] A B. Andreica, "Implementing Parameterized Type Algorithm Definitions in Mathematica", Eighth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Lisa O'Connor editor, IEEE Computer Society Press, 2006, pp. 1-6.

[4] B. Buchberger, "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory", in Mathematics and Its Applications, Multidimensional Systems Theory, N. K. Bose Ed., D. Reidel Publishing Co., 1985, chapter 6.

[5] B. Buchberger, "Gröbner Bases and Systems Theory", Multidimensional Systems and Signal Processing, 12, Kluwer Academic Publishers, 2001, pp. 223-251.

[6] B. Buchberger, "Towards the Automated Synthesis of Groebner Bases Algorithm", RACSAM, vol Falta, 2004, pp.1-10.

[7] C W. Cobb and P H Douglas, "A Theory of Production", American Economic Review, 18, 1928, pp. 139-165.

[8] D. Gruntz and M. Monagan, "Introduction to Gauss", MapleTech, Birkhuser, 9, 1993, pp. 23-35.

[9] D. Gruntz, "Gröbner Bases in Gauss", MapleTech, Birkhuser, 9, 1993, pp. 36-46.

[10] K. Iwancio and M. Singer, "Applications of Groebner Bases", http://www4.ncsu.edu/~kmiwanci/app_gbases_2.pdf [retrieved: 05, 2013]

[11] V. Levandovskyy, "Non–commutative Computer Algebra for polynomial algebras: Groebner bases, applications and Implementation", http://d-nb.info/976594358/34 [retrieved: 05, 2013]

[12] V. Powers and B. Reznick, "A new bound for Pólya's Theorem with applications to polynomials positive on polyhedra", Journal of Pure and Applied Algebra, Vol. 164, Issues 1–2, Oct 2001, pp. 221–229 http://www.sciencedirect.com/science/article/pii/S0022404900001559 [retrieved: 05, 2013]

[13] C. Ratiu-Suciu, F. Luban, D. Hincu, and N. Ene, Modelarea si simularea proceselor economice , Biblioteca electronică a Academiei de Studii Economice Bucureşti, [retrieved: 05, 2013]
http://www.ase.ro/biblioteca/carte2.asp?id=70&idb=7

[14] B. Sandelin, "The Early Use of Wicksell-Cobb-Douglas Function: A Comment on Weber", Journal of the History of the Economic Thought, Vol. 21, Issue 02, Cambridge University Press, June 1999, pp. 191-193, doi: http://dx.doi.org/10.1017/S105383720000314X.

[15] S. Wolfram, Mathematica, 1992.

[16] Mathematica Online Tutorial [retrieved: 06, 2013] http://reference.wolfram.com/mathematica/guide/Mathematica.html

[17] G. Zaman, Z. Goschin, I. Partachi, and C. Herteliu, "The Contribution of Labour and Capital to Romania's and Moldova's Economic Growth", Journal of Applied Quantitative Methods, Vol. 2, Issue 1, March 30, 2007, http://jaqm.ro/issues/volume-2,issue-1/pdfs/zaman_goschin_partachi_herteliu.pdf [retrieved: 06, 2013]

[18] Capital flows and labour costs values for countries in 2013 http://www.tradingeconomics.com/country-list/capital-flows, [retrieved: 09, 2013]