

MATLAB as an Instrument for Reduction of Payoff Matrix in Games with Non-zero Sum

Zuzana Čičková¹, Allan Jose Sequeira Lopez²

Abstract

Generally, the game theory is used for analysis of conflicting decision making situations (games) with multiple subjects (players) involved. The simplest games include two-player games where the payoffs for both players can be easily characterized in a matrix form. Although a notation of this type of game is relatively simple, the possibility of finding equilibrium strategies is rather complicated. The related mathematical model is the nonlinear programming problem. The aim of the paper is to point out the possibilities of reducing payoffs matrices to considerably simplify the solution of the respective game. The reduction will be performed by the iterative procedure in the MATLAB software.

Key words

Bimatrix Games, Matrix Reduction, Game Theory, MATLAB

JEL classification

C71, C61, C790

1 Introduction

The paper is focused on payoff matrix reduction in case of two-player games (bimatrix games). Mathematically, bimatrix games can be formulated as nonlinear programming problems. Solving such problems is generally complicated, so it is logical to try to reduce the size of corresponding problem as much as possible. In the case of bimatrix games, it is sometimes possible to reduce the individual players' payoff matrix. Illustrative examples of matrix reduction will be performed in the system MATLAB³.

2 Brief Characteristics of Basic Game Types

Bimatrix game is a game with two participants (players). Each player chooses independently (without information about the choice of an opponent) one of the final number of behavioral variations (strategies). It is assumed that the players' interests are not diametrically opposed, i.e. the profit of one of the players does not necessarily means the loss of the other. It is also assumed that the players are intelligent. The question is what strategy the player has to choose so that choosing another strategy cannot increase his profit. A bimatrix game can be formalized as follows: Let $P = \{1, 2\}$ be a set of players, each player has a finite set of strategies (X – player 1, Y – player 2), i.e. player 1 chooses $x \in X$, player 2 chooses $y \in Y$. A set of all the game results can be labeled as $(x, y) \in X \times Y$. The elements of set X and Y can be arranged with the finite number of natural numbers (elements of the set X : $i = 1, 2, \dots, m$

¹ doc. Ing. Zuzana Čičková, PhD., University of Economics in Bratislava, Faculty of Economic Informatics, Department of Operations Research, and Econometrics, Dolnozemska cesta 1/b, 852 35 Bratislava, cickova@euba.sk.

² Ing. Allan Jose Sequeira Lopez, University of Economics in Bratislava, Faculty of Economic Informatics, Department of Operations Research, and Econometrics, Dolnozemska cesta 1/b, 852 35 Bratislava, allan.lopez@euba.sk.

³ The calculations were performed on terminal server ISS CVTI SR.

and elements of the set $Y: j = 1, 2, \dots, n$). The game's values for the player 1 can be written in a matrix $\mathbf{A}_{m \times n} = \{a_{ij}\}$, where a_{ij} indicates the payoff of player 1 at the result (i, j) . The results of the game for the player 2 can be written in the matrix $\mathbf{B}_{n \times m} = \{b_{ji}\}$, where b_{ji} indicates the payoff of player 2 at the result (i, j) .

Bimatrix games can be formulated as non-linear programming problem with a generally complex solution. In order to find equilibrium in bimatrix games, there are various algorithms. One of them is a method of describing submatrices of \mathbf{A} and \mathbf{B} yielding all extreme points of a set of equilibrium solution (Kuhn, 1961). Other methods reduce the problem of finding equilibrium solutions of a bimatrix game to a problem of quadratic programming (Lemke & Howson, 1964). The problem of linear programming serving to identify the equilibrium points not satisfying the Karusha-Kuhna-Tucker optimality conditions is shown, in (Čičková & Zagiba, 2018). The following part of the paper is focused on the ways of reducing payoff matrix that can lead to a substantial simplification of the game's solution.

3 Reasons and Possibilities of Payoff Matrix Reduction

A general description of the matrix reduction procedure can be described in following steps:

1. Set $j=1$ (here “j” being a “row counter”, and so we are starting from the first row)
2. Rearrange rows $j, j+1, \dots, n$ so that the leading entry of row j is positioned as far to the left as possible.
3. Multiply row j by a nonzero constant to make the leading entry equal 1.
4. Use this leading entry of 1 to reduce all other entries in its column to 0 using elementary row operations.
5. If any rows $j+1, \dots, n$ contain nonzero terms, increase j by 1 and go to step number 2.

For the purpose of our analysis we will use the following statement for reduction of payoff matrices: The payoff matrix reduction is based on the following statement:

If in a game with payoff matrix $\mathbf{A} = (a_{ij})$ type $m \times n$ applies to some $k, 1 \leq k \leq m$ and for all j the following relation

$$a_{ij} \geq a_{kj}, i \neq k, \quad (1)$$

then there is an optimal mixed strategy of the player 1 $\mathbf{x}^{(0)}$ with component $x_k^{(0)} = 0$.

At the same time, the argument can be extended by considering linear combination (Chobot et al., 1991).

If in a game with payoff matrix $\mathbf{A} = (a_{ij})$ type $m \times n$ it is true that for some $k, 1 \leq k \leq m$ there are such numbers $p_i (i \neq k)$, that for all $j = 1, 2, \dots, n$ we get

$$\sum_{i \neq k} p_i a_{ij} \geq a_{kj}, \sum_{i \neq k} p_i = 1, p_i \geq 0, \quad (2)$$

then there is an optimal mixed strategy of the player 1 $\mathbf{x}^{(0)}$ with component $x_k^{(0)} = 0$.

Obviously, the analogical conclusion also applies to the reduction of the matrix $\mathbf{B}_{n \times m} = \{b_{ji}\}$.

The relations (1) and (2) define so-called weakly dominant strategies. Generally the following applies: equilibrium of the game with weakly dominant strategies is also equilibrium in the original game as well. However, this process may delete other equilibrium from the game. Unfortunately, there is no way of knowing whether some equilibrium is removed. For this reason it is generally recommended to use strict dominant. We can identify this kind of strategies on the basis of relationships:

$$a_{ij} > a_{kj}, i \neq k, \quad (3)$$

$$\sum_{i \neq k} p_i a_{ij} > a_{kj}, \sum_{i \neq k} p_i = 1, p_i \geq 0 \quad (4)$$

Based on the above mentioned the conclusion is as follows If the k -th row of payoff matrix of the game H is dominated by a convex linear combination of other rows of this matrix, then such row can be struck out from the payoff matrix and then the game H_I can be solved with such reduced matrix. Optimal mixed strategies of the player 2 in the game H_I are his optimal strategies in game H as well. Extending at k -th point of each optimal mixed strategy of player 1 in the game H_I is his optimal strategy also in the game H . The values of game H and H_I are the same.⁴ Reduction in MATLAB in the fourth chapter is presented for strictly dominated strategies but a little change in the program code can do it also for weak dominated strategies.

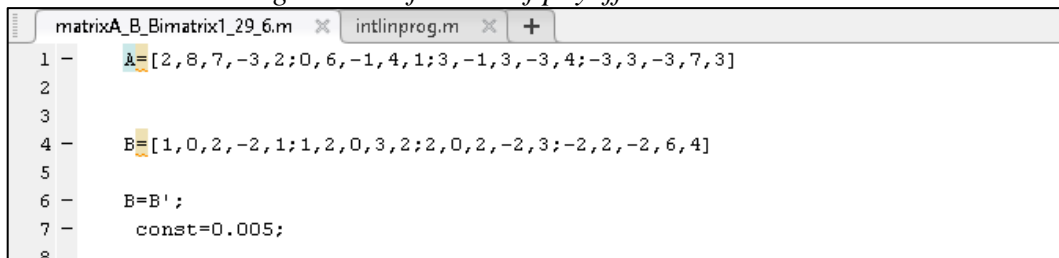
4 Payoff Matrix Reduction in MATLAB

The MATLAB name was created by shortening the words MATrix LABoratory, which corresponds to the fact that matrix data is the key data structure for MATLAB calculations. The programming language is based on Fortran (Dušek, 2002). MATLAB is a programming environment with its own programming language that specializes in scientific numerical computing, modeling, algorithm design, computer simulations, data analysis and presentation, signal measurement and processing, control and communication system designs.

The possibilities of payoff matrix reduction in a bimatrix game will be illustrated by the following example.

Let's assume the following payoff matrix and also initialize other program parameters. The matrix A and B represent the payoff matrices of both players. These are matrices type $m \times n$ with 5 column and 4 rows. Then we have to define a constant - we have chosen a constant of 0.005 used for testing linear combinations (for strictly dominant strategies)⁵ in both matrices **A** and **B** (Figure 1 and 2).

Figure 1: Definition of payoff matrix A and B



```

1 - A=[2,8,7,-3,2;0,6,-1,4,1;3,-1,3,-3,4;-3,3,-3,7,3]
2
3
4 - B=[1,0,2,-2,1;1,2,0,3,2;2,0,2,-2,3;-2,2,-2,6,4]
5
6 - B=B';
7 - const=0.005;
8

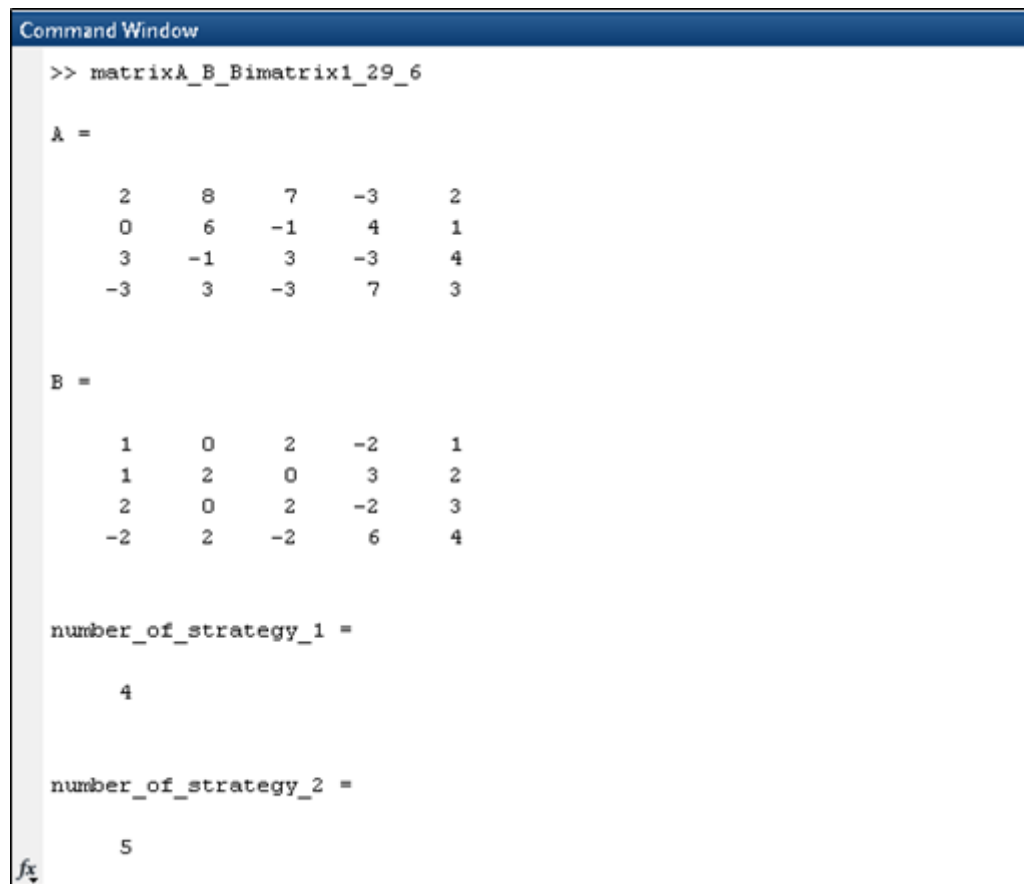
```

Source: Own representation

⁴ Sometimes only sharp dominant is used ($>$), because at weak dominant (\geq) would result to the solution where not all equilibrium points are identified (Čemická & Čičková, 2011).

⁵ In the case of weak dominant the constant is set at 0.

Figure 2: Definition of payoff matrix A and B



```

>> matrixA_B_Bimatrix1_29_6

A =

     2     8     7    -3     2
     0     6    -1     4     1
     3    -1     3    -3     4
    -3     3    -3     7     3

B =

     1     0     2    -2     1
     1     2     0     3     2
     2     0     2    -2     3
    -2     2    -2     6     4

number_of_strategy_1 =

     4

number_of_strategy_2 =

     5

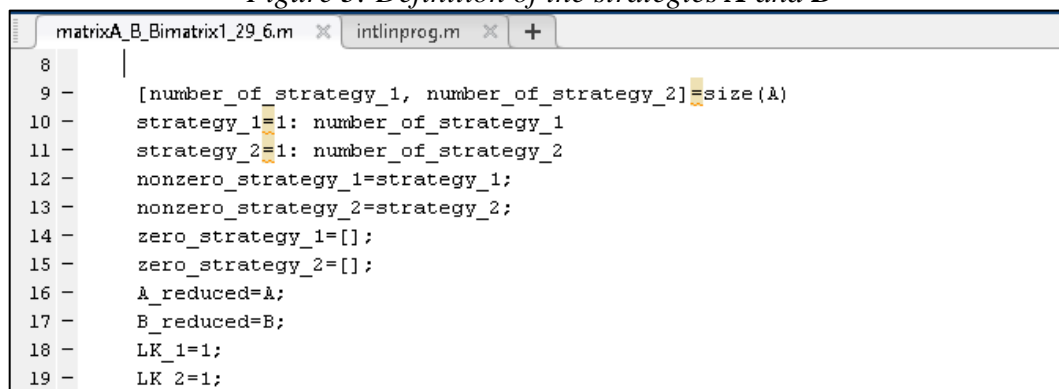
```

Source: Own representation

As we see at the (Figure 2) the number of strategy of the first player is four and the number of strategy of the second player is five.

After that we define the number of possible strategies for player 1 and player 2.

Figure 3: Definition of the strategies A and B



```

matrixA_B_Bimatrix1_29_6.m x intlinprog.m x +
8 |
9 - [number_of_strategy_1, number_of_strategy_2]=size(A)
10 - strategy_1=1: number_of_strategy_1
11 - strategy_2=1: number_of_strategy_2
12 - nonzero_strategy_1=strategy_1;
13 - nonzero_strategy_2=strategy_2;
14 - zero_strategy_1=[];
15 - zero_strategy_2=[];
16 - A_reduced=A;
17 - B_reduced=B;
18 - LK_1=1;
19 - LK_2=1;

```

Source: Own representation

The basic reduction of the matrices can be done as follows. In the program code, a cycle (While Loop) was created to find the rows and columns dominated by others. The command has the following structure:

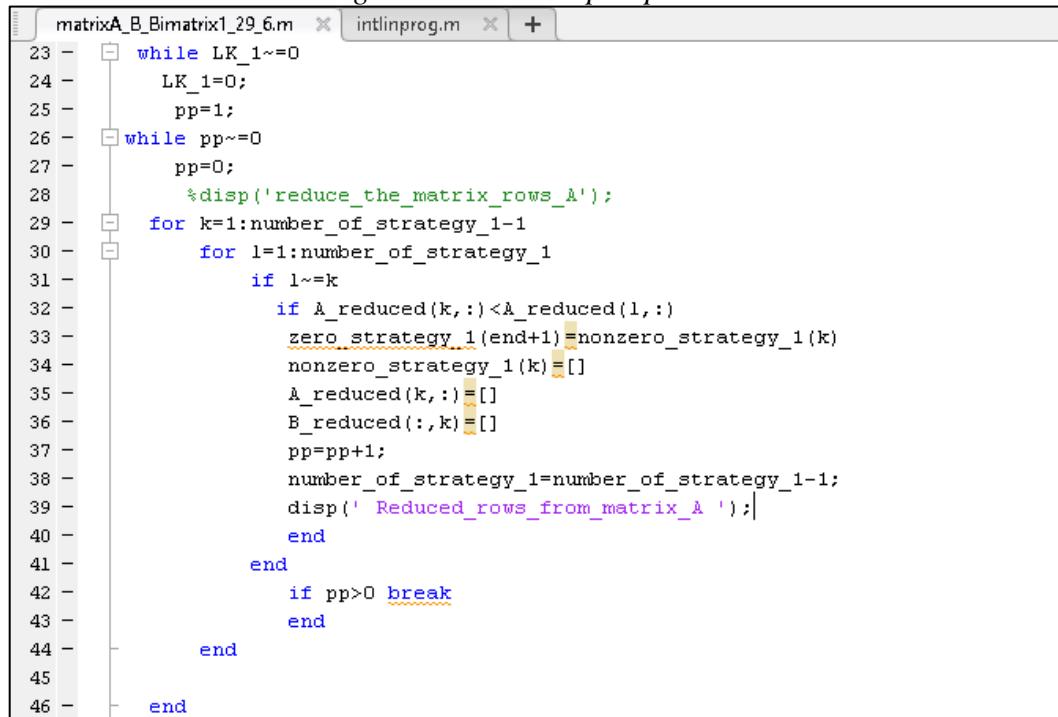
while expression

statements

end

The command evaluates the expression and repeats the execution of a command group in cycles if the expression is true. The expression is true when its result is non-empty and contains only non-zero elements (logical or real numeric). Otherwise, the expression is untrue.

Figure 4: While Loop Expression



Source: Own representation⁶

In the code, we can see the first part of While Loop – by using command “for” we define the area and parameters for which the program is to perform the cycle; by using command “if” we set the conditions to validate the calculation as well as conditions the program shall consider the task to be terminated.

The cycle in MATLAB is ended by command “break” if the searched condition is satisfied. The command “end” is used to finish the whole cycle so we can proceed with the code by entering a new command. Similarly we can perform the reduction of matrix **B** (Figure5).

⁶ Changing the relation from < to <= in the line 32 we eliminate also the weak dominant strategies of matrix **A**

Figure 5: Reduction of Matrix B

```

matrixA_B_Bimatrix1_29_6.m  intlinprog.m  +
47  %disp('reduce_the_matrix_rows_B');
48  for k=1:number_of_strategy_2-1
49      for l=1:number_of_strategy_2
50          if l~=k
51              if B_reduced(k,:) < B_reduced(l,:)
52                  zero_strategy_2(end+1)=nonzero_strategy_2(k)
53                  nonzero_strategy_2(k)=[]
54                  B_reduced(k,:)=[]
55                  A_reduced(:,k)=[]
56                  pp=pp+1;
57                  number_of_strategy_2=number_of_strategy_2-1;
58                  disp('Reduced rows from matrix B');
59              end
60          end
61      end
62      if pp>0 break
63  end
64  end
65
66  end
67  end
68  end

```

Source: Own representation

This part of the code returns the reduced matrix **B**. So far, we only used the assumption (3), however, it is also necessary to examine the linear combinations based on the relationship (4). In case of application of the abovementioned rule, the result is as follows:

Figure 6: Reduction of Matrix B

```

Command Window

Optimal solution found.

A_reduced =

     8     7    -3     2
     6    -1     4     1
    -1     3    -3     4
     3    -3     7     3

B_reduced =

     0     2     0     2
     2     0     2    -2
    -2     3    -2     6
     1     2     3     4

x =

    0.2525
    0.2475
         0
    0.5000

```

Source: Own representation

In order to investigate the linear convex combinations of all elements in the code, we open the MATLAB toolbox for linear programming – linprog

`[x,fval,exitflag,output,lambda] = linprog (c,matrix,a,R,r,lb,ub,[],[])`,

the left side returns an output structure and the right side minimizes the optimization parameters specified in the structure options. The last parameters are optional (for more information see e.g. Čičková, 2015).

The input parameters are characterized in the Table 1 and the output parameters are characterized in the Table 3.

Table 1: Input parameters

Input Parameter	Characteristics
C	vector of coefficients of objective function
A	matrix of structural coefficients related to inequation
a	the right side of the inequation type boundary
B	matrix of structural coefficients corresponding to equations
b	the right side of the equation type boundary
Lb	vector of the lower boundaries of decision variables
Ub	vector of the upper boundaries of decision variables
x0	variable vector of start values, if not known, then []
Options	the options to set parameters using the function optimset.m

Source:www.mathworks.com (4.7.2018) and (Čičková, 2015)

The tool „options“ allows setting several managing parameters, e.g. by using the function optimset.m with the following structure:

`options=optimset('ParameterName1',value1,'ParameterName2',value2,...)`

The list of most common parameters used for solving the problems of linear programming is included in the Table 2.

Table 2: Chosen parameters of function optimset.m

Parameter	Values
'LargeScale'	'on','off'
'Simplex'	'on','off'
'Display'	'iter','final','off'
'Maxiter'	Maximum amount of iteration

Source:www.mathworks.com (4.7.2018) and (Čičková, 2015)

Solver linprog.m has implemented the following three algorithms:

- primary-dual interior point algorithm – preset algorithm
- simplex algorithm – can be set by turning off the primary-dual interior point algorithm using `options=optimset('LargeScale','off')`
- active matrix algorithm – can be set by turning off the simplex algorithm `options=optimset('Simplex','off')`

Table 3: Output parameters

Output Parameter	Characteristic
X	vector of optimal variable values
Fval	Values of objective function
Exitflag	a parameter that determines whether the algorithm converges (exitflag > 0), or not
Output	a structure providing information about the number of iterations, the type of algorithm used, etc.
Lambda	a structure providing information about Lagrange multipliers

Source: www.mathworks.com (4.7.2018) and (Čičková, 2015)

Next, we will present the use of linprog to reduce the dominated strategies in MATLAB. We can see all structure in for matrix **A** in Figure 7.

Figure 7: Test of Linear Combination of Matrix A

```

matrixA_B_Bimatrix1_29_6.m  x  intlinprog.m  x  +
71 -   disp('I try to find the linear combinations in matrix A');
72 -   for k=1:number_of_strategy_1-1
73 -       matrix= A_reduced;
74 -       a=-matrix(k,:)+const;
75 -       matrix(k,:)=[];
76 -       matrix=-matrix';
77 -       lb=zeros(1,number_of_strategy_1-1);
78 -       ub=lb+1;
79 -       R=ones(1,number_of_strategy_1-1);
80 -       r=1;
81 -       c=zeros(1,number_of_strategy_1-1);
82 -       [x,fval,exitflag,output,lambda]=linprog(c,matrix,a,R,r,lb,ub,[],[]);
83 -       if exitflag>=0
84 -           A_reduced=-matrix'
85 -           B_reduced(:,k)=[]
86 -           x
87 -           zero_strategy_1(end+1)=nonzero_strategy_1(k)
88 -           nonzero_strategy_1(k)=[]
89 -           number_of_strategy_1=number_of_strategy_1-1;
90 -           LK_1=LK_1+1;
91 -           end
92 -           if number_of_strategy_1==2
93 -               break
94 -           end
95 -       end
96 -   end

```

Source: Own representation

All structure for matrix **B** is shown on Figure 8. From rows 72 -122 in (Figure 7) and (Figure 8), the conditions were preset to find all linear combinations between rows in matrix **A** and matrix **B** in MATLAB using the Linprog toolbox. The final solution of illustrative example is reduced matrices for both players; at first the software displays the payoff matrix and the number of strategy for both players. (Figure 8).

Figure 8: Test of Linear Combination of Matrix B

```

matrixA_B_Bimatrix1_29_6.m  intlinprog.m  +
97  %disp('I try to find the linear combinations in matrix B');
98  for k=1:number_of_strategy_2-1
99      matrix= B_reduced;
100     a=-matrix(k,:)+const;
101     matrix(k,:)=[];
102     matrix=-matrix';
103     lb=zeros(1,number_of_strategy_2-1);
104     ub=lb+1;
105     R=ones(1,number_of_strategy_2-1);
106     r=1;
107     c=zeros(1,number_of_strategy_2-1);
108     [x,fval,exitflag,output,lambda]=linprog(c,matrix,a,R,r,lb,ub,[],[]);
109     if exitflag>=0
110         A_reduced(:,k)=[];
111         B_reduced=-matrix';
112         x
113         zero_strategy_2(end+1)=nonzero_strategy_2(k);
114         nonzero_strategy_2(k)=[];
115         number_of_strategy_2=number_of_strategy_2-1;
116         LK_2=LK_2+1;
117     end
118     if number_of_strategy_2==2
119         break
120     end
121 end
122 end

```

Source: Own representation

Figure 9: Elimination of Dominated Strategies

```

Command Window

Optimal solution found.

A_reduced =

     7     -3     2
    -1     4     1
     3     -3     4
    -3     7     3

B_reduced =

     2     0     2    -2
    -2     3    -2     6
     1     2     3     4

x =

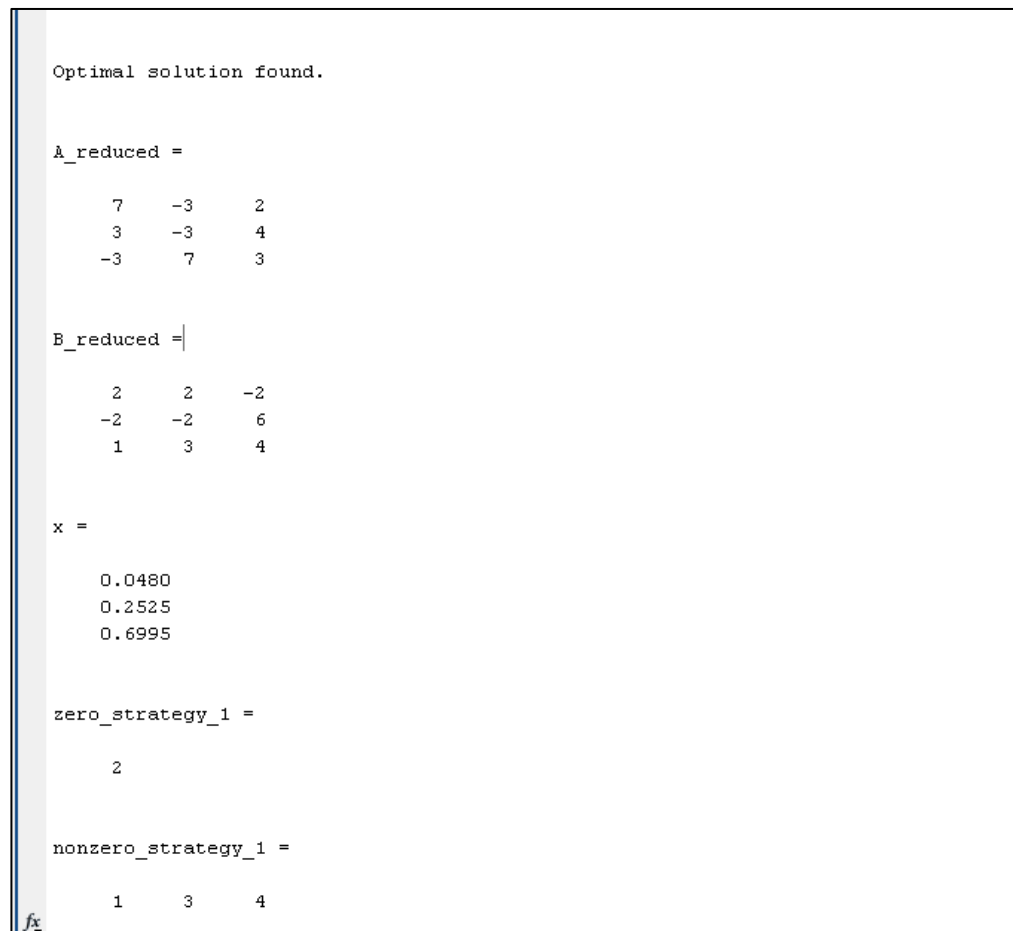
    0.2040
    0.4030
    0.3930

```

Source: Own representation

As shown above (Figure 9) after strategy definition, the software tries to find all dominated strategy and eliminate them. Then, the software tries to find linear combinations for both matrices (Figure 10).

Figure 10: Solution – Finding Linear Combinations and Reduction of Matrices



Source: Own representation

5 Conclusion

Matrix reduction is undoubtedly an interesting tool for simplifying input parameters of bimatrix games because generally even in the case of two-player games a nonlinear problem is to be solved. The solution of a such kind of problem is usually very complicated and therefore there is an aim to find every possibility to make it easier.

The practical use of the reduction requires adequate software tools. In the paper, the authors presented the code for the reduction created in the MATLAB system. The authors were able to solve the problem in a hypothetical example. The code is generally applicable to bimatrix and matrix game. With a small change the code can be used to solve a problem with more than two players.

Acknowledgements

The paper was prepared as part of the project VEGA 1/0351/17 Application of Selected Models of Game Theory in Solving Some Economic Problems in Slovakia.

Reference

- [1] Čemická, K., & Čičková, Z. (2011). Teória hier a aukcie na Slovensku a ich porovnania s ostatnými vo svete. In *Nové trendy v ekonometrii a operačnom výzkumu : Mezinárodní vědecký seminář* (pp. 34-28). Prague, CZ: Katedry ekonometrie FIS VŠE v Praze. doi:978-80-225-3317-1
- [2] Čičková, Z. (2015). Použitie Optimization Toolbox v MatLabe. In *Mladá veda AIESA – budovanie spoločnosti založenej na vedomostiach* (16th ed., Vol. 1, pp. 559-564). Bratislava, SC: EKONÓM. doi:978-80-225-4151-0
- [3] Čičková, Z., & Zagiba, M. (2017). Podmienky Optimálnosti Karusha-Kuhna Tuckera a Bimaticové Hry. In *Aplikácia vybraných modelov teórie hier pri riešení niektorých ekonomických problémov Slovenska*(Vol. 1, pp. 1-7). Bratislava, SC: EKONÓM. doi:978-80-225-4430-6
- [4] Dušek, F. (2000). Matlab Simulink úvod do používání [PDF]. Pardubice: Univerzita Pardubice.
- [5] Chobot, M., Turnovec, F., & Ulašín, V. (1991). *Teória hier a rozhodovania* (3rd ed., Vol. 1, Ser. 2). Bratislava, SC: Alfa. doi:80-05-00702-7
- [6] Kuhn, H. (1961). An Algorithm for Equilibrium Points in Bimatrix Games. *Proceedings of the National Academy of Sciences of the United States of America*, 47(10), 1657-1662. Retrieved from <http://www.jstor.org/stable/71305>
- [7] Lemke, C. E., & Howson, Jr, J. T. (1964). Equilibrium Points of Bimatrix Games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2), 413-423. doi:<https://doi.org/10.1137/0112033>.
- [8] MathWorks. (2017). *mathworks.com*. (MathWorks, Inc) Cit. 2017. Dostupné na Internet: https://www.mathworks.com/solutions.html?s_tid=gn_sol
- [9] Moler, C. (2004). *The Origins of MATLAB*. MathWorks. Dostupné na Internet: Technical Articles and Newsletters: <https://ch.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html>