Transport and Communications, 2020; Vol. I.
ISSN: 1339-5130

The Use of Monte Carlo Simulation Method in Decision Support Via MATLAB Script

Viliam Mojský[1], Karol Achimský[1]

[1]Department of communications, Faculty of Operations and Economics of Transport and Communications, University of Žilina, Univerzitná 1, 010 26 Žilina, Slovakia

Abstract In our research we examined the possibility of using Monte Carlo simulation to support decision making through simulation models. For this purpose, we chose a basic economy task assignment. Subsequently, we created a simulation algorithm in Matlab, where we searched for the optimal amount of offered products based on the task inputs. The output of the research is a simulation algorithm using the Monte Carlo method to solve a given economic problem.

Keywords simulation methods, Monte Carlo, Matlab, decision support

JEL C63

1. Introduction

Simulation allows us to simulate the behaviour of the real system via a created model. Using this model, it is possible to simulate various events that may occur. Based on the simulation results, it is possible to estimate with some probability the development of the real situation. Simulation and simulation models are powerful tools that can help us with decision making, with the right inputs. The correctness of these models is influenced by the correct determination of the relationships, functions and probabilities of events to be simulated.

In the research we dealt with the Monte Carlo simulation method. We applied this method to an example from the field of economics in which the method was used as a tool to support decision making between several eventualities. We used Matlab to create the simulation model.

2. Materials and Methods

The chapter briefly describes the theoretical background and the program we used to create the simulation.

2.1. Simulation, model, Monte Carlo

Simulation is the process of creating a model of a real system and conducting experiments with it to achieve a better understanding of the behaviour of the system studied, or to assess various variants of system. A system is a set of elements that are organized in a certain way. A real system is a part of the real world that is of interest. We can refer to the model as a guide or a system that provides, to some extent, the same behaviour as a real system [1,2].

The term model is used to refer to both material and immaterial imitation of an object, regardless of the purpose for which it is used. The imitation itself can be implemented with practical tasks, it can fulfil certain functions, or it can be used for research purposes [3,4].

The Monte Carlo method is called a numerical method of solving mathematical problems by modelling random variables and statistical estimation of their characteristics. The Monte Carlo method can be used to solve any mathematical problems, both stochastic and deterministic [5].

The Monte Carlo method is a numerical method based on the relationship between the probability characteristics of various random processes and the quantities that are the solution of the studied problems [6].

2.1. Matlab

Matlab is a programming platform for research and scientific purposes. It is based on Matlab's own programming language. It is a matrix-based language that allows to naturally express mathematical calculations. Matlab combines a working environment for iterative analysis and process design with a programming language that directly expresses matrices and mathematical fields. It is possible to analyse data, create algorithms and create models and applications using Matlab [7].

Matlab has several working environments. We have chosen Live Script. It is a new work environment that combines Matlab code with formatted text and image output in the Live Editor environment [8].

3. Results

At the beginning of the chapter is given an economic problem, for which we created a simulation model by Monte Carlo method in Matlab. After the task assignment, the flowchart and the individual parts of the simulation model are presented.

3.1. Task assignment

The post office manager is facing an uneasy task in the pre-Christmas period. Before Christmas, postcards with the wishes of "Merry Christmas and a Happy New Year" are sold well as ancillary goods. One postcard is purchased by the postal manager for 0.70 Eur and sold for 1.20 Eur. Postcards that are not sold by the end of the New Year period are then sold at a 50% discount. The difficult task of the postal manager is to decide how many postcards to order in order to maximize the profit from postcard sales. If the demand for postcards is higher than the quantity ordered, the post office will lose profits due to lack of postcards. Conversely, if the demand for postcards is lower than the quantity ordered, the post office will lose some of the profits due to the Christmas discount sale.

The profit equation if demand is greater than supply is:

$z = (1{,}20 – 0{,}70) * q$

If demand is smaller than supply, then the profit equation is:

$z = 1{,}20 * d – 0{,}70 * q + 0{,}60 * (q – d)$

The simulation model inputs are:

- Ordered quantity q (decision variable),
- Different income levels and expenditure factors (constants)
- Demand d (probability variable)

If demand is known, we can calculate the profit from the equations above. Because demand is probabilistic variable, we must choose the value of the demand from probability distribution of demand. We have defined the decision problem by assuming a level of demand from the range of 400-900, from which we will randomly select numbers using the randi function. This selects pseudo-random numbers from the given range according to an even distribution.

3.2. Flowchart

We have created a development diagram according to the given task assignment. The flowchart is shown in Figure 1.

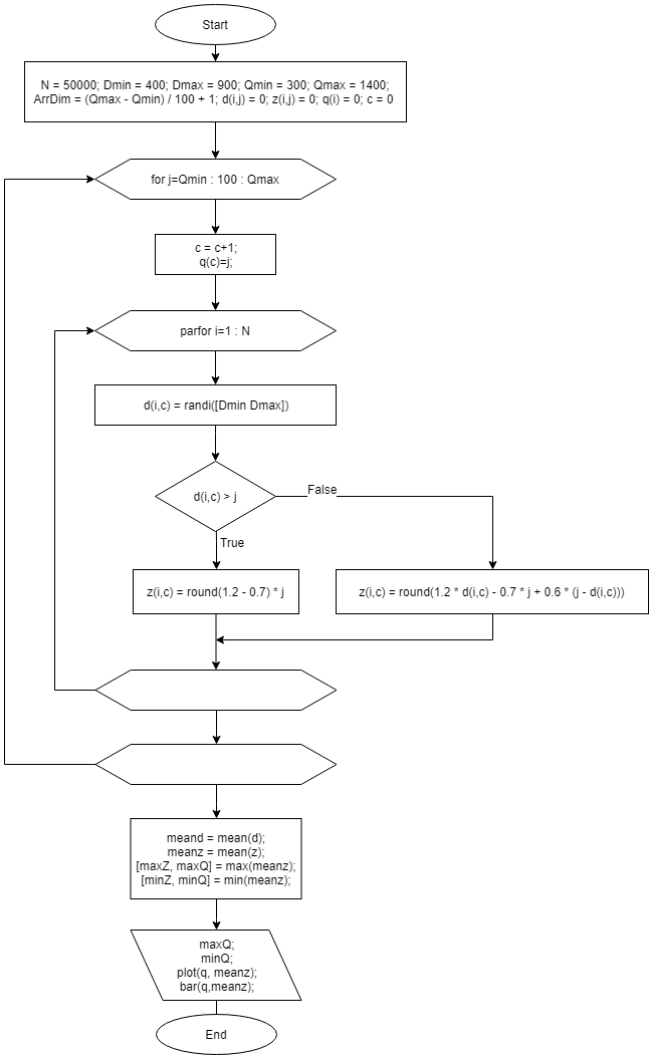

Figure 1. Algorithm flowchart

3.3. Simulation model in Matlab

Based on the flowchart, we started writing the algorithm code in Matlab. We created the code in Live Script. This allowed us to use some additional features, such as displaying outputs without opening additional program windows. Another advantage is that we can run the program on any computer through the Matlab cloud environment without the need to install Matlab.

The created program was divided into three logical parts. We decided for the three parts because they logically divided the program into inputs, algorithm operations and outputs. These parts are described separately in their own chapters. Parts of the source code used are listed in each chapter.

3.3.1. Initialization part

In initialization part are defined input variables, which are used in the program.

Source code:

```
clear;
tic;
```

DOI: 10.26552/tac.C.2020.1.4


```
N = 50000;
Dmin = 400;
Dmax = 900;
Qmin = 300;
Qmax = 1400;

ArrDim=(Qmax-Qmin)/100+1;
d = zeros(N,ArrDim);
z = zeros(N,ArrDim);
q(:,1) = Qmin:100:Qmax;
c = 0;
rng(47);
```

The clear function flushes the cache and all values from previous instances of this or another Matlab script. This ensures that past results do not affect the current instance. In addition to biasing the results, without flushing the cache, bugs could also be generated as a result of using variables from previous scripts. The clear function is followed by the tic function. This is a paired function, the pair to which is the toc function, located at the end of the source code. The tic toc function is used to measure the time required to execute a program. The measured time is displayed in the output window along with the other outputs. The function can be used multiple times in the program to determine which part of the program is the most time consuming and needs optimization. The function was mainly used during development. In the final version, it is used to compare how changing the size of input values affects the length of program execution.

Input variables follow. N denotes the number of repeats, i.e. the number of generated pseudo-random numbers that will be used in one iteration of the cycle. The value is set to 50,000, which we consider sufficient. Dmin and Dmax, which represent the minimum and maximum predicted dose, were also determined. They are used in the program as the lower and upper bounds of the range from which pseudo-random numbers are generated. Qmin and Qmax were set after them to set the minimum and maximum quantities offered. These are used in the first cycle as the minimum demand, to which 100 is added at each iteration until the maximum demand is reached.

After defining the input variables, the initialization section also contains commands for pre-allocation of variables. This is done to speed up the implementation of the program. In the program, fields are filled at each cycle iteration. Matlab can work faster with a field if it does not have to extend it each time it is filled, but simply overwrites the value at given field position. The number of columns in the array is initially determined. This value is stored in the ArrDim variable and is calculated by subtracting the maximum and minimum demand, dividing the result by 100 and adding 1 to it. Adding 1 is required because Matlab does not work with arrays like other programming languages. The initial value in the two-dimensional array has a coordinate of 1.1, unlike other programming languages, where the first coordinate value is 0.0. The result of this calculation is used for three successive populations of the variables d and z. Both are defined as two-dimensional arrays with dimensions N (number of rows) and ArrDim (number of columns). Generated predicted query is stored in variable d. The variable z stores the calculated profit based on the expected demand. Both variables are filled with zeros that are overwritten when the program executes. Each level of the ordered quantity has its own column in which the measured values are stored. Another variable defined is q. This is filled with levels of offered quantity from minimum to maximum, increasing by 100. Since we know that variable q will be a 1-dimensional vector, we have defined it as a column array with (:, 1) statement, instead of the default row array. We did this because of faster access to its values, because Matlab can work faster with column variables. This is an optimization approach in declaring variables. Next, the variable c has been defined to be zero. This variable provides the addressing of the correct column in the "d" and "z" fields used in the program. The last command is the rng function. This provides a seed for pseudo-random numbers, so that every time the simulation is repeated, we always get the same result. It is set to 47, a nonnegative integer that generates predictable pseudo-random numbers. If we always want a different result, we can set the value ‘shuffle‘.

3.3.2. Generating pseudo-random numbers

The next part of the program lists the cycles that ensure that a specified number of pseudo-random numbers will be generated. In the cycles there are also formulas from the example, which are placed in a decision statement, which, based on the input parameters, decides which of the formulas is used.

Source code:

```
for j=Qmin:100:Qmax
    c=c+1;

    parfor i=1:N
        d(i,c)=randi([Dmin Dmax]);
        if d(i,c)>j
            z(i,c)=(1.2-0.7)*j;
        else
          z(i,c)=1.2*d(i,c)-0.7*j+0.6*(j-d
          (i,c));
        end
    end

end
```

At the beginning of the code is a for loop. This ensures that the cycle runs in a predetermined number of iterations. In this case, it ensures the execution of orders inside, for each level of ordered quantity separately. It starts with the minimum order quantity and increases in 100-increments up to the maximum order quantity. This is followed by a command that increases the value of c. This ensures that each level of ordered quantity will have its own column index. This is followed by the parfor command. This is a

parallel for command. Parallelization means that the command can be executed in parallel, or simultaneously, with other commands. This is a very useful method that can significantly speed up program execution. By default, the program is run on a single thread. Multiple processor threads can be utilized through parallelization. This means that on multi-core and multi-threaded processors, the program will run significantly faster. However, we need to be very careful about what processes we parallelize, because improper use can cause errors in the program, which may not occur immediately during testing. The parfor cycle runs N times for i from 1 to N. In our case, N is 50,000, so the cycle runs 50,000 times. It is important that the variables that have been produced in the parfor are indexed. This prevents errors such as overwriting variables. For example, a pseudo-random number d (i, c) could cause a serious program error without an index. This is because every new iteration of the cycle generates a new pseudo-random number. If the program runs on 4 threads, the number d would be generated 4 times. Since generating is the first operation within a cycle, the last generated number d would be stored in variable d. This number would continue to be used in the cycle and could be changed again while running because the threads are working at the same time. In this case, the results would be significantly distorted and rendered irrelevant. Therefore, the variable d is indexed, ensuring that each iteration of the cycle has its own number d, to which it assigns its own pseudo-random number with which the cycle works and no further iteration can change it. It is important that the index changes with the cycle iteration. If it were changed by calculation, it could lead to errors again. In this case, the index changes according to the variable i, which is defined in the parameters of the parfor cycle, which ensures its security. The body of the cycle also contains the variable j, which does not have an index. However, this is fine, this variable is invariant for all iterations of the parfor cycle. Its value is determined by the parent for loop. In the parfor cycle, it is used only in calculations, which is logically correct. The variable c is similar. The last variable in the parforum cycle is the variable z (i, c), which again changes its value at each iteration. Its calculation depends on the result of the decision order. This variable is also indexed to avoid errors in the program.

Within the parfor cycle, pseudo-random numbers are generated and evaluated. The profit is then calculated. The first command generates a pseudo-random number via the randi function. This generates numbers from the range given by the minimum and maximum predicted demand, according to the even distribution. The generated number is stored in the variable d (i, c). Subsequently, the number generated is equal to the level of the ordered quantity j via the if statement. If the result of the comparison is true, the generated number is greater than the quantity ordered, i. the demand is larger than the offer, then the if statement is executed. If the result of the comparison is not true, so the offer is greater than the demand, then is executed the else branch. Both formulas in the if statement are taken from the assignment. Their task is to calculate the profit based on the result of the comparison of demand with offer. The calculated gain is stored in the variable z (i, c). The profit calculation is followed by the end of the if decision statement and both cycles are followed by end keywords.

3.3.3. Algorithm outputs

Outputs from the model are divided into two parts, namely text part and graphic part. Text outputs generates sentences indicating maximum and minimum values. Graphical outputs contain several variants of possible Matlab outputs.

Text outputs

Source code:

```
meanz(:,1)=round(mean(z),2);
[maxZ,maxQ] = max(meanz);
[minZ,minQ] = min(meanz);

odpoved1 = ['Najväčší zisk bude pravdepodobne
dosiahnutý pri ponúkanom množstve
',num2str(q(maxQ)), 'ks vo výške '
,num2str(maxZ),'€.'];
disp(odpoved1);
odpoved2 = ['Najmenší zisk bude pravdepodobne
dosiahnutý pri ponúkanom množstve
',num2str(q(minQ)), 'ks vo výške '
,num2str(minZ),'€.'];
disp(odpoved2);
```

At the beginning of the code is defined the meanz variable, which is a one-dimensional array. Again, to optimize performance, it is defined as a column vector by the (:, 1) statement. This variable stores the calculated average value for each column from the profit field of (i, j). The results are rounded to two decimal places. In the next two commands, its maximum and minimum (maxZ and minZ) as well as their position in the array (maxQ and minQ) are determined from the meanz field. Writing the generated variables in square brackets is necessary because the value and position of the search value in the array is determined.

In the answer1 statement, a text response is formulated by string interpolation. The variables had to be modified before being inserted into the sentence. These are numbers, so they are of type number and must be converted to a string with the num2str function to be inserted into a sentence. The first number is read from the field q (i), where "i" is replaced by the detected maxQ variable, which represents the position of the maximum quantity. The second number is the maxZ variable, which is also converted to a string. The disp function is used after the generated responses to display the generated sentences. For minimum values, the same procedure was used, but with different values. The resulting outputs are shown in Figure 2.

Graphic outputs

Source code:

```
xaxmin = Qmin-Qmin*0.2;
```

DOI: 10.26552/tac.C.2020.1.4


```
  xaxmax = Qmax+Qmax*0.05;
  yaxmin = minZ-minZ*0.2;
  yaxmax = maxZ+maxZ*0.1;

plot(q,meanz,'-s','MarkerSize',5,'MarkerEdg
eColor','red','MarkerFaceColor',[1 .6 .6]);
hold off;
axis([xaxmin xaxmax yaxmin yaxmax]);
ylabel('Priemerný zisk');
xlabel('Q');
plot(q,meanz,'-s','MarkerSize',5,'MarkerEdg
eColor','red','MarkerFaceColor',[1 .6 .6]);
hold off;
axis([xaxmin xaxmax yaxmin yaxmax]);
text(q,meanz,num2str(meanz'),'vert','bottom
','horiz','center');
ylabel('Priemerný zisk');
xlabel('Q');
plot(q,meanz,'color','#0072BD');
hold on;
stem(q,meanz,'.','color','#0072BD');
axis([xaxmin xaxmax yaxmin yaxmax]);
text(q,meanz,num2str(meanz'),'vert','bottom
','horiz','center');
hold off;
ylabel('Priemerný zisk');
xlabel('Q');
bar(q,meanz);
text(q,meanz,num2str(meanz'),'vert','bottom
','horiz','center');
box off;
ylabel('Priemerný zisk');
xlabel('Q');

toc;
```

Four graphs were created within the graphical outputs to illustrate the possibilities of graphical outputs from the Matlab program.

There are 4 variable declarations at the beginning of the code. These were created to modify the boundaries of the x and y axes in the graphs. This step is not necessary, the program would draw graphs without it. By default, however, the graphs starts at 0.0, the maximum of the y-axis is the maximum value of y and the maximum of x is the maximum of x, so the points are drawn at the graph boundaries, which is disturbing, especially if we want to add a description or highlight them. Therefore, we have declared 4 points that determine the minimum and maximum on the x and y axes. Their name is an abbreviation of their purpose, e.g. xaxmin is the x axis minimum, or the minimum on the x axis. Their values are calculated based on a given value, e.g. the maximum on the x-axis, plus 5% of its length. The others are calculated similarly, but with other coefficients, which we determined by observing the graph.

Declaring the borders is followed by the creation of the first graph. The command to create a line chart is plot. This is a very simple command in which only the first two parameters, x and y values, are required. Other parameters are set by default. So, at the beginning we have x and y axes. We set the first parameter of the x-axis to the variable q and the second parameter of the y-axis draws data from the meanz variable. Subsequently, the '-s' statement defines the line type and point markings on the graph. The line will be solid '-' and the points on the graph will be marked with a square 's' (square). Next we defined other parameters of the graph. There are no rules of order when defining them. Thus, any parameter can be defined anywhere within the bracket of the plot command. Therefore, it is always necessary to first define the parameter that can be set by its name. The parameter name is followed by the parameter value. The first parameter that was defined was the size of the point mark on the graph using the 'MarkerSize' keyword. We set the value to 5 units. 1 unit = 1/72 inch. Next, we set the border color of the point marker to red using the 'MarkerEdgeCo-lor keyword and the' red 'value. The color was set directly by its name, which is defined in Matlab. Next, we set the color of the interior of the point on the graph to a faint red shade with the keyword 'MarkerFaceCo-lor' and the value [1 .6 .6]. It is an abbreviated color definition notation according to the RGB scheme [1 0.6 0.6]. We deliberately used two different kinds of color definition to show that it is possible to use multiple kinds of definitions. The plot command is followed by a hold command with the off parameter. The hold command determines whether the next chart is rendered, whether to the current chart or to a new chart window. Off means no, so the graph will not be held, and the new graph will be plotted in the new window. These two commands alone are enough to generate graphs. We added to them the modification of the x and y axes borders and their names. The axis command sets the boundaries of the x and y axes. We used the values calculated at the beginning of the section. The following are the ylabel and xlabel commands that add names to the x and y axes. The first graph output is shown in Figure 2.

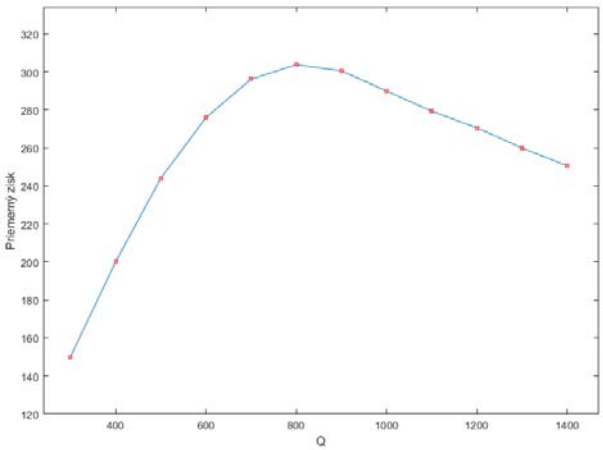

Figure 2. The first graph output.

We defined the second graph as the first, with one difference. We have added names to the chart using the text

function to add descriptions to the chart at the specified points. Similar to plotting a graph, the x and y axes need to be specified in this function to know where to place text markers. We used the variables q and meanz on the x and y axes. Another parameter in the function is the text of the axis description. Here we entered meanz values. However, we had to convert them to a string because number values cannot be output. The num2str function was used for this. In addition, inside the function, the upper quotation mark ‘was used to separate meanz values. Without the quotation mark, the entire meanz field would be displayed at each point. The quotation mark ensures that it displays only the corresponding value, the others are cut off. That would be all the mandatory parameters, but we also set the vertical and horizontal alignment. Vertical alignment with the ‘vert‘ keyword to ‘bot-tom’ or down. Horizontal alignment with the ‘horiz‘ keyword to ‘center‘. The other parameters of the graph are the same as the first. The output of the second graph is shown in Figure 3.

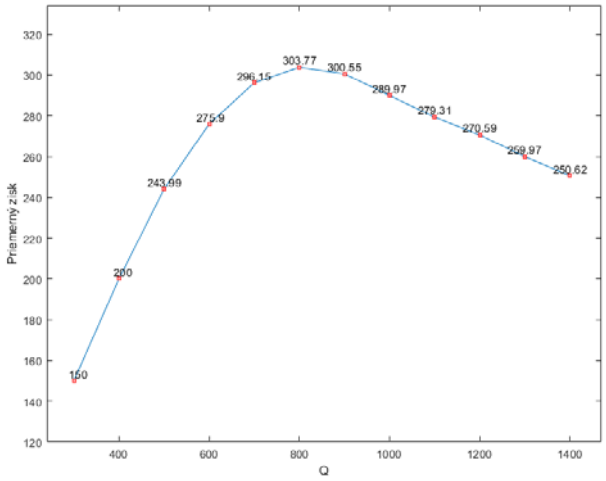

Figure 3. The second graph output.

The third statement again extends the previous statement. First, it contains two hold functions. The first is set to on, the second is set to off. This is because between them there is a plot of the second type of graph - the vertical line graph, that plots into the body of the previous graph. The first hold on function allows us to draw a second graph into the first body. The stem function was used to draw a line graph. Its input parameters are the same as the plot function, except for the output display. Instead of a line graph where the line connects the points formed by the coordinates, it plots vertical lines from the x-axis to the height determined by the y-axis. In the function parameters, we set the line type to dashed by the ‘-‘ statement. We have turned off the point marking with the ‘Marker‘ parameter, which we set to ‘none‘, i.e. no marking. The colour of the chart was set by the ‘color’ parameter, which we assigned a value in the form of the hexadecimal colour code‘ # b3d9ff ‘. We chose this method to illustrate, along with the previous examples, the possibilities of defining colours and to show that there are several right paths to the result. Again, the graph function is followed by definitions of boundaries, text, and description of the graph, and a hold function with the parameter off to prevent another graph from being drawn into the body. Boundaries, texts and graph descriptions are sufficient if two graphs are drawn into one output only once, it is not necessary to do it for each. The output of the third graph is shown in Figure 4.

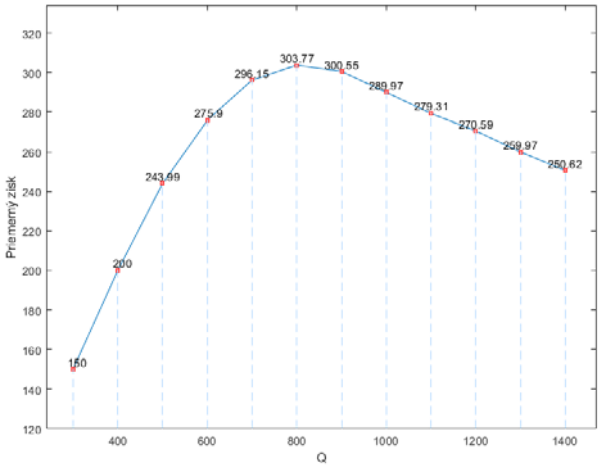

Figure 4. The third graph output.

The last, fourth graph is of the column type. To create it, use the bar function, which again accepts the same parameters as the previous chart types. In this case, we decided to define only the x and y axes with the variables q and menaz. We left the other parameters as default. The following are statements to display descriptions above the graph columns, naming the x and y axes. These commands are the same as in the previous graphs. In addition, there is a function box with parameter off. The default is on, so on. The box creates a bounding line around the graph with a scale from all sides. Since the x and y axes have their own scaling and have a value above each column, we decided to turn this feature off. The output is shown in Figure 5.

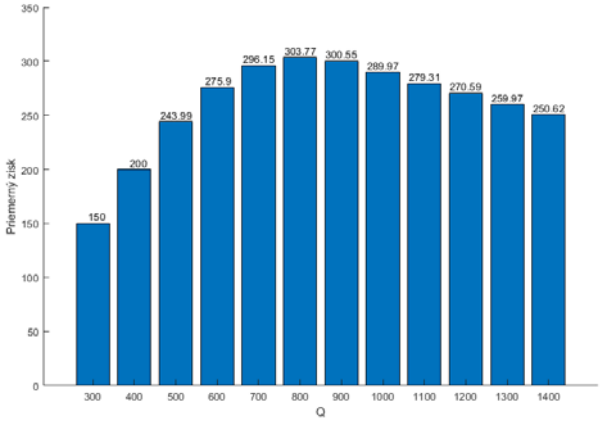

Figure 5. The fourth graph output.

Behind the graph creation code is the toc function described above to record the execution time of the program. This is the end of the algorithm's source code.

DOI: 10.26552/tac.C.2020.1.4

In addition to the listed outputs, a Workspace is also available. Workspace is a space where all variables created during the program are stored. They are available after the execution is complete, or during the execution (unless clear statement is used). All variables can be exported as a table in multiple formats, e.g. .txt, or .xls through the writetable () function. This data can be used for further analysis and to generate additional outputs. The workspace created in our program is shown in Figure 6. It contains an overview of all the variables that have occurred. Variables labelled as a multiplication of two numbers are vectors indicating their dimensions.

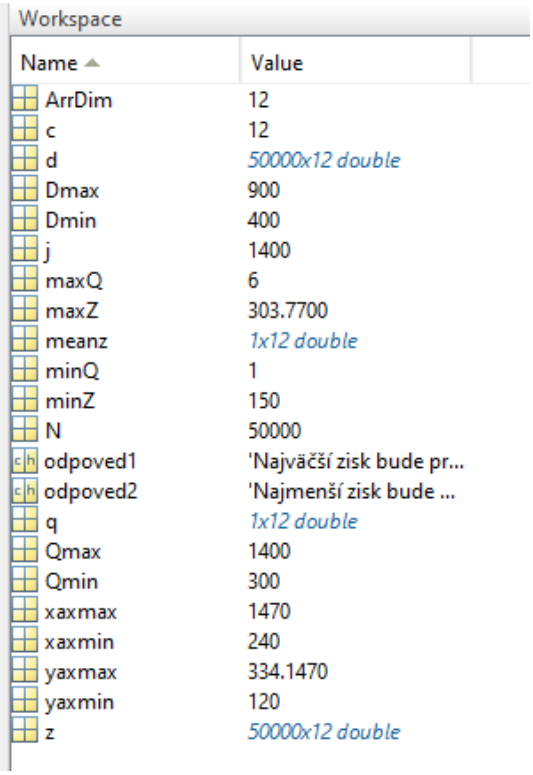

Figure 6. Workspace with variables.

3.4. Simulation results

Through the created simulation model, we can proceed to the solution of the given example. Inputs and input formulas are shown in the described simulation model. We decided to draw the conclusions from the last type of bar graph shown in Figure 5, because it seems to us most readable.

In the simulation, the examined quantity of goods was added. We set it from 300 to 1400 pieces of postcards. We decided for a minimum of 300 because the minimum demand is 400 and we wanted to point out the difference in profit. In the case of 300pcs and 400pcs, there could never have been a discount sale and the products were sold at full margin. Therefore, the difference in profit between the two options is the largest of the two options. Full purchasing power has always been used. At 500 pieces, a slight decrease in the rate of profit growth is evident. For other supply volumes, the rate of increase in profit gradually decreases. The maximum is reached at the level of 800 postcards. This amount achieved an average estimated profit of EUR 303.77. Beyond the supply level of 800, the average profit started to decline. The graph in Figure 5 shows that the rate of descent is less than the rate of climb to the maximum.

By extending the ordered quantity to 2400 pieces and leaving the same amount of attempts to generate pseudo-random numbers 50,000 and other parameters, we get the graph shown in Figure 7.

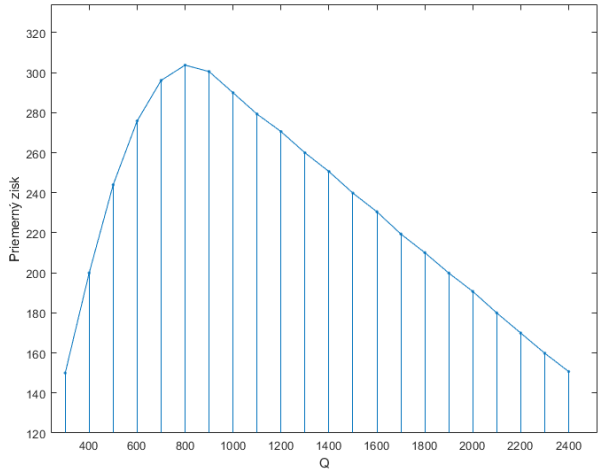

Figure 7. Extended graph

From the graph it can be concluded that values from 300Q to 800Q have a distribution similar to the square root function, while values from 800Q to 2400Q are more like a linear function distribution.

Conclusion

By this research we wanted to show the possibility of using Monte Carlo method to support managerial decision making through simulation model. We created a model that was applied to a simple economic example from the postal environment. The model itself was created in Matlab to show the possibilities of this program. Through the model, the demand of customers was simulated and the optimal amount of products offered was found, with which according to the established formulas the greatest profit will be achieved. Based on the results of the simulation, we found that the biggest profit will be achieved by the company with the offered quantity Q = 800 products in the amount of EUR 303.77.

The created model provides the user with various types of outputs in both text and graphical form. The user also has stored variables to work with and analyse in order to obtain additional outputs. The advantage of the model is its modularity. In the algorithm we used a uniform distribution to generate pseudo-random numbers. This distribution can be replaced by another distribution by modifying a command

in a program that provides the generation of elements into the variable d (i, c).

ACKNOWLEDGEMENTS

This article was published with the support of project EUREKA-E! 11158 U Health Auto-ID technológie a internet vecí na zvýšenie kvality zdravotníckych služieb.

This article was published with the support of project VEGA 1/0721/18 Výskum ekonomických dopadov vizuálneho smogu v doprave s využitím metód neurovedy.

REFERENCES

[1] R. Hušek, J. Lauber. Simulačné modely. Praha: SNTL/ALFA, 1987. 349s. ISBN 978-80-562-0075-9

[2] Achimský, K., Čorejová, T., Fitzová, M. Kajánek, B. Projektovanie sietí v pošte I. Vysoká škola dopravy a spojov v Žiline. Edičné stredisko VŠDS, Žilina. 1995. 147s. ISBN 80-7100-238-0

[3] Všeobecné základy modelovania. [online]. [citivané 25.05.2019]. Dostupné na internete: http://www.fbi.uniza.sk/ktvi/leitner/2_predmety/OA/00_Vseobecne%20o%20modelovani.pdf

[4] Achimský, K. Simulácia s použitím jazyka GSAP II. In: R-687-010 ˇUVT VŠDS. Žilina. 1982. 38s.

[5] Achimská. V. Modelovanie systémov. Žilina: Žilinská univerzita v Žiline, 2011. 96 s. ISBN 978-80-554-0450-9

[6] Úvod do modelovania a simulácie, metóda Monte Carlo. [online]. [citované 25.05.2019]. Dostupné na internete: http://fbi.uniza.sk/ktvi/leitner/2_predmety/SMOA/04_StatistikaExcel/MCprednaska.pdf

[7] What is Matlab? [online]. [citované 26.05.2019]. Dostupné na internete: https://nl.mathworks.com/discovery/what-is-Matlab.html

[8] What Is a Live Script or Function? [online]. [citované 26.05.2019[. Dostupné na internete: https://nl.mathworks.com/help/Matlab/Matlab_prog/what-is-a-live-script-or-function.html