Faculty of Sciences and Mathematics, University of Niš, Serbia
Available at: http://www.pmf.ni.ac.yu/filomat

Filomat **23:2** (2009), 68–81

VISUALIZATION IN OPTIMIZATION WITH MATHEMATICA

Predrag S. Stanimirović*, Marko D. Petković, Milan Lj. Zlatanović

Abstract

We show how the computer algebra system in MATHEMATICA and its graphical capabilities can be used in optimization. A package for teaching the graphical solution of two-dimensional and three-dimensional linear programming problem is developed.

1 Introduction

It is well-known that MATHEMATICA is the premier software system for numerical, symbolic, and graphical computations and visualization [9, 10, 11]. It is a very high level programming language, adaptive to various types of courses in mathematics. MATHEMATICA is a system for doing mathematics by computer. It includes arbitrary precision and exact numerical computation, symbolic computation, graphics, sound, hyperlinked documentation and interprocess communication - all integrated together in one easy-to-use package. The MATHEMATICA computer algebra system is an attractive medium for teaching mathematics [1]. There exists several computer algebra systems, such as Maple, MatLab and others, and at this level it is mainly a matter of taste which system one chooses to use. However, looking a little further it is the authors opinion in [4] that MATHEMATICA is superior to the others for the following reasons: MATHEMATICA as a programming language is very structured and highly adaptive to a variety of applications, both technical and theoretical.

In teaching programming for students in mathematics courses, one of the important features for programming languages may be the ability to treat functions as higher order functions. This feature is presented in [6] and compared with programming language C.

Let us notice that there are no standard packages for visualization and animation of linear programming problems and the geometrical method in the package MATHEMATICA. Furthermore, there is no such implementation, even in the most complete MATHEMATICA optimization software, described in Bhatti [2]. Therefore, the

*Corresponding author

2000 *Mathematics Subject Classifications.* 90C05, 68W30.

Key words and Phrases. Linear programming; Visualization; Computer graphic; Computer Algebra System; MATHEMATICA

first goal of this work is to alleviate this deficiency. The second aim of the paper is to extend possibilities of the programming language MATHEMATICA. With respect to this purpose, the paper is intended to provide research and/or development results.

In the present paper we are concerned with the application of MATHEMATICA's symbolic computation and graphical capabilities in optimization theory. Optimization can be done in many different programming languages such as FORTRAN, C++, and specialty languages. MATHEMATICA is a high level programming language that offers many advantages for optimization. Optimization capabilities in MATHEMATICA are investigated in [5]:

- Very high precision math is standard.
- A huge library of advanced math functions is available.
- The notebook user interface is easy to use and interactive.
- Most critically for optimization, symbolic manipulation of expressions is possible.

Main disadvantage of MATHEMATICA is the slow execution. While the slow execution of the code (inherent to all interpreted programming languages and MATHEMATICA) is a clear disadvantage, code execution is only 1.% of the total research time. In any research, most time is spent reading, thinking and trying out the ideas. In the computer languages that lack the visualization tools, or file import/export conversion tools, researchers are forced to spend weeks, if not months, implementing those features only at a barely functioning level. MATHEMATICA on the other hand will allow the researchers to focus on the research most of the time.

We investigate symbolic and graphical capabilities of package MATHEMATICA in optimization. In the second section we investigate visualization and animation of $2D$ and $3D$ linear programming problems, and describe a MATHEMATICA software for teaching the graphical method for solving two-dimensional and three-dimensional linear programming problems. Our software consists of two independent modules Geom2D and Geom3D which are designed for visualization of 2D and 3D linear programming problems. Complete software is written and fully functional in MATHEMATICA 6.0. Some pieces of code remained compatible with MATHEMATICA 5.2. There are some compatibility issues of running program in the last version MATHEMATICA 7.0, but there are supplied appropriate alternatives.

2 Animation of geometrical method in linear programming

The set of points satisfying constraints of the form $A\mathbf{x} \leq b$, $A \in \mathbb{R}^{m\times n}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} \geq 0$ can be represented as the intersection of a finite number of closed half-spaces. Thus, these constraints define a convex polytope. We assume, for simplicity, that this polytope is nonempty and bounded. In other words, the equations of constraints define a polyhedron Ω_p in $\mathbb{R}^n$. Let H be a hyperplane of support of this polyhedron. If the dimension of Ω_p is less than n, then the set of all points common to the hyperplane H and the polyhedron Ω_p coincides with Ω_p. If the dimension of Ω_p

is equal to n, then the set of all points common to the hyperplane H and the polyhedron Ω_p is a face of the polyhedron. If this face is $(n-1)$-dimensional, then there exists only one hyperplane of support, namely, the carrier of this face. If the dimension of the face is less than $n-1$, then there exists an infinite number of hyperplanes of support whose intersection with this polyhedron yields this face [3]. The goal of our linear programming problem is to maximize a linear objective function $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = c_1x_1 + \cdots + c_nx_n$ on the convex polyhedron Ω_p.

The set of points where the goal function $f(\mathbf{x})$ have a constant value d represents a hyperplane $H_{f,d} = \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) = d\}$, which can be translated in the direction of the vector $\mathbf{c}$, varying the value d. The linear programming problem is equivalent with the problem of finding a maximal (minimal) value d satisfying $H_{f,d} \cap \Omega_P \neq \emptyset$. It is well known that the extreme value of goal function is achieved in one of the extreme points of the set Ω_P or in each point on the edge of the set Ω_P which is parallel with the straight line $f(\mathbf{x}) = d$. This method is applicable in cases $n = 2$ and $n = 3$.

2.1 2D case

Consider the the general form of linear programming (LP) problem in $\mathbb{R}^2$:

$$\begin{array}{lll} \max \quad f(\mathbf{x}) & = & f(x,y) = c^T\mathbf{x} = c_1x + c_2y \\ \text{s.t.} & & a_{i1}x + a_{i2}y \leq b_i, \quad i \in I_1 \\ & & a_{i1}x + a_{i2}y \geq b_i, \quad i \in I_2 \\ & & x, y \geq 0. \end{array} \tag{2.1}$$

where $I_1 \cup I_2 = \{1, \ldots, m\}$, $I_1 \cap I_2 = \emptyset$ and a_{ij}, b_i, c_j are given real numbers and $m = |I_1| + |I_2|$. Each inequality constraint from (2.1) determines a subset $D_i \subset \mathbb{R}^2$, $i = 1, \ldots, m$, representing the set of points on the one side of corresponding straight line $a_{i1}x + a_{i2}y = b_i$. Therefore, the set of feasible solutions (polyhedron denoted as Ω_P in $\mathbb{R}^2$) is determined as the intersection

$$\Omega_p = D_1 \cap D_2 \cap \cdots \cap D_m \cap D_{m+1} \cap D_{m+2},$$

where subsets D_{m+1}, D_{m+2} of $\mathbb{R}^2$ are derived from the conditions $x \geq 0, y \geq 0$.

Since the objective function in (2.1) is of two variables, it can be applied well known geometrical procedure for solving the linear programming problems [7]. If the restricting conditions in (2.1) are given in the form of inequalities, each of the corresponding straight lines divides the area into a range which is possible for these conditions and a range impossible for these conditions. The permissible conditions are located in the range Ω_P that is permissible for all conditions (the region of feasible solution). The optimal solution is found by drawing the graph of the modified objective function $f(x,y) = 0$ and parallel shifting of this in the direction of the gradient vector (c_1, c_2). The optimal solution is unique if the straight line $f(x,y) = f_{max}$ runs trough a corner point of the possible range. In minimization, the straight line must be shifted in the opposite direction. Animation of this two

dimension case consist of parallel shifting of the line $f(x, y) = 0$ across the feasible region.

Function `Geom2D` uses the following parameters:

1. f: the goal function;
2. g: the list of given constraints.

Algorithm corresponding to $2D$ case consists the next three steps.

Step 1. Visualization of the mathematical model (2.1) in linear programming uses the following graphical primitives:

1. the region of feasible solutions, denoted by *region*;
2. vertices obtained as intersections of boundary hyperplanes in Ω_P, denoted by *points*;
3. text corresponding to generates vertices, denoted by *txt*;
4. the objective function line, denoted by *objective*.

First we plot the constraints set (graphics *constraints*) applying the standard MATHEMATICA function `InequalityPlot`, contained in the package `Graphics‘InequalityGraphics`. It is done by the following piece of code:

```
var = Variables[f];
region = InequalityPlot[g, {var[[1]]}, {var[[2]]},
                          AspectRatio->1,DisplayFunction->Identity
         ];
```

Solving given inequality constraints we get the feasible set Ω_P, denoted by h. For this purpose we use standard function `InequalitySolve` from the package `Algebra‘InequalitySolve`.

```
h = g /. {List -> And}; h = InequalitySolve[h, var];
```

Note that the packages `Graphics‘InequalityGraphics` and `Algebra‘InequalitySolve` are obsolete in `MATHEMATICA 7.0`. Instead of the functions `InequalityPlot` and `InequalitySolve`, in `MATHEMATICA 7.0` there should be used `RegionPlot` and `Reduce` respectively. Code for `h` remains the same while the code for `region` becomes:

```
p2 = RegionPlot @@ {And @@ g, {var[[1]], 0, mx}, {var[[2]], 0, my},
AspectRatio -> 1};
```

and should be moved after the code for computing `mx` and `my`.

In the following for loop we find all extreme points lying on the edges the feasible region, using the standard function `FindInstance`. These extreme points are stored in the list *res*, and they are obtained as intersections of each two straight lines obtained after the transformation of constraints given as inequalities in equalities, replacing headers *LessEquaL, Equal, GreaterEqual, Greater* by the function *Equal*.

```
g1 = g/.{LessEqual->Equal, GreaterEqual->Equal, Less->Equal,
            Greater->Equal};
For[i = 1, i <= Length[g1] - 1, i++,
  For[j = i + 1, j <= Length[g1], j++,
    t = FindInstance[g1[[i]] && g1[[j]] && h, var];
    If[t != {},
     AppendTo[res, {t[[1, 1, 2]], t[[1, 2, 2]]}];
     AppendTo[res2, {ReplaceAll[f, t[[1]]], t[[1]]}];
     ];
    ];
  ];

res2 = Union[res2]; res = Union[res];
```

Points contained in the list *res* are plotted on the graphics *points*.

```
points = ListPlot[res, PlotStyle -> {PointSize[0.025], Hue[1]}];
```

The graphics primitive *txt* displays text of the form A_i centered at the point specified by the expression $res[[i]] + res[[1]]/7$:

```
txt = Graphics[Table[{Text["A" (Subscript[i]), res[[i]]+res[[1]]/7]},
                        {i,Length[res]}
                    ]
      ];
```

The graphics primitive *objective* denotes the line corresponding to the equation $f == 0$:

```
objective = Solve[f == 0, var[[2]]][[1, 1, 2]];
```

These graphical primitives are plotted using the following code:

```
mx = Max[Table[res[[i, 1]], {i, 1, Length[res]}]];
my = Max[Table[res[[i, 2]], {i, 1, Length[res]}]];

plot = Show[region, points, txt,
   TextStyle -> {FontFamily -> "Times", FontSize -> 14},
   AxesLabel -> TraditionalForm /@ var,   AspectRatio -> 1,
   PlotRange -> {{-mx/10, mx*1.1}, {-my/10, my*1.1}}];
```

Step 2. Animation of geometrical procedure for solving the linear programming problem (2.1). The animation consists of the following table of graphical primitives:

(a) graphical primitives denoted by $region, points$ and txt;

(b) straight lines $objective + n$, where the variable n runs through the feasible region.

The plotted graphics are animated using the function `Manipulate`:

```
For[i = 1, i <= Length[res], i++,
    n = ReplaceAll[var[[2]] - objective, res2[[i, 2]]];
  ];
n1 = ReplaceAll[var[[2]] - objective, res2[[1, 2]]];
n2 = ReplaceAll[var[[2]] - objective, res2[[Length[res2], 2]]];

Manipulate[Show[plot,
     Plot[objective + n, {x, -mx/10, mx*1.1},
          PlotStyle -> {Thickness[0.007], RGBColor[0, 0, 1]}]
    ], {n, n1, n2}] // Return;
```

Note that in MATHEMATICA 6.0 and 7.0 the same code works also with function `Animate` instead of the function `Manipulate`. In MATHEMATICA 5.2 we need to load the external package `Graphics‘Animation‘` in order to use the function `Animate`.

Step 3. Program gives optimal solution, in the case of its uniqueness; otherwise, it produces the expression which describes the set of all optimal points.

```
If[res2[[Length[res2],1]]==res2[[Length[res2]-1,1]],
      Print["Optimal solution is given by \[Lambda]*",
      res[[Length[res]]],"+(1-\[Lambda])*",res[[Length[res]-1]],
       ",    0<=\[Lambda]<=1"],
      Print["Optimal solution is: ",res[[Length[res]]]];
    ];
```

Example 1. Applying the expression

```
Geom2D[8x+12y,{8x+4y<=600,2x+3y<=300,4x+3y<=360,5x+10y>=200,
                          y<=80+x,y>=x-40,x>=0,y>=0}]
```

we obtain the next animation:

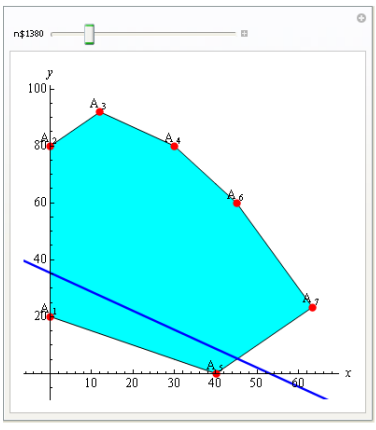

Figure 1. Snapshot of the animation given by Example 1.

Program gives a set of graphics which contain graphical illustration of the feasible set Ω_P together with its vertices, text corresponding to vertices and the graphical representation of the straight line $H_{f,d}$. Straight line $H_{f,d}$ translates upward in the direction of the vector $c = \{8, 12\}$ across the feasible set Ω_p while the condition $H_{f,d} \cap \Omega_p \neq \emptyset$ is satisfied.

In this case, optimal points are defined by the expression

$$\text{Optimal solution is given by } \lambda^* \left\{ \frac{190}{3}, \frac{70}{3} \right\} + (1\text{-}\lambda)^* \{45, 60\},\ 0 \leq \lambda \leq 1.$$

2.2 3D case

Consider the general form of the linear programming problem in $\mathbb{R}^3$ ($3D$ problem):

$$\begin{array}{lcll} \max & f(\mathbf{x}) & = & f(x,y,z) = c_1x + c_2y + c_3z \\ \text{s.t.} & & & a_{i1}x + a_{i2}y + a_{i3}z \leq b_i, \quad i \in I_1 \\ & & & a_{i1}x + a_{i2}y + a_{i3}z \geq b_i, \quad i \in I_2 \\ & & & x, y, z \geq 0. \end{array} \tag{2.2}$$

Similarly as in the $2D$ case, the set of feasible solutions (polytope in $\mathbb{R}^2$ denoted by Ω_P) is determined as the intersection

$$\Omega_p = D_1 \cap D_2 \cap \cdots \cap D_m \cap D_{m+1} \cap D_{m+2} \cap D_{m+3},$$

where $D_i \subset \mathbb{R}^3$ is set of the solutions (half-space) of the i-th inequality in (2.2) and D_{m+1}, D_{m+2}, $D_{m+3} \subset \mathbb{R}^3$ are derived from the conditions $x \geq 0, y \geq 0, z \geq 0$.

The standard geometrical procedure from [7], already explained in the subsection 2.2, can also be applied to solve the problem (2.2). Although, we will use slightly different approach in order to visualize this geometrical method. Corresponding algorithm is implemented `MATHEMATICA` function `Geom3D` which solves given $3D$ problem and gives the interactive visualization of the geometrical procedure. This function has the following parameters:

1. c: the goal function;
2. $cond$: the list of the constraints;
3. $\{x, y, z\}$: the list of the variables containing three elements.

Similarly to the previous $2D$ case, algorithm consists of the following major steps:

Step 1. Visualization of the set Ω_P of feasible solutions.

Feasible solution set Ω_P of the $3D$ problem (2.2) is a convex polytope in $\mathbb{R}^3$, in general case. This polytope is visualized by drawing its boundary. For this purpose we used built-in `MATHEMATICA` 6.0 function `ContourPlot3D`. Boundary of the polytope Ω_P can be characterized as the set of the solutions of inequalities from

(2.2) such that at least one inequality holds with equality sign. In other words, let us take j-th inequality from (2.2) and treat is as the equality:

$$a_{j1}x + a_{j2}y + a_{j3}z - b_j = 0. \tag{2.3}$$

It is not important whether $j \in I_1$ or $j \in I_2$. Consider the set of all solutions $(x, y, z) \in \Omega_P$ of system (2.2) such that equality (2.3) holds. Denote this set with $(\partial\Omega_P)_j$. Boundary $\partial\Omega_P$ of the set Ω_P is obtained as the union of all $(\partial\Omega_P)_j$ for every $j \in I_1 \cup I_2$. In other words, it holds:

$$\partial\Omega_P = \bigcup_{j \in I_1 \cup I_2} (\partial\Omega_P)_j. \tag{2.4}$$

From the last expression, it holds that the set Ω_P can be drawn by drawing its boundary, i.e. sides $(\partial\Omega_P)_j$ for every $j \in I_1 \cup I_2$. To draw a side $(\partial\Omega_P)_j$ we will use built-in MATHEMATICA function ContourPlot3D. It is done by the following code

```
feas=ContourPlot3D@@@{surf,{x,0,xmax},{y,0,ymax},{z,0,zmax},
RegionFunction->Function@@{{x,y,z},region},
BoxRatios->{1,1,1},Mesh->None,Axes->False,PlotPoints->30};
```

Function ContourPlot3D plots the surface given by the equation of the form $F(x, y, z) = 0$ where $F(x, y, z)$ is known function. The syntax of ContourPlot3D is the following:

```
ContourPlot3D[f,{x,xmin,xmax},{y,ymin,ymax},{z,zmin,zmax}]
```

Here f is the expression representing the function $F(x, y, z)$ and *xmin, xmax, ymin, ymax, zmin, zmax,* determine the region where surface will be plotted. To plot more than one surface on the same graph, parameter f should be the list of the functions (expressions) corresponding to the surfaces. In order to plot $(\partial\Omega_P)_j$ using ContourPlot3D, parameter f should be given by the left hand side of the equation (2.3). Actually, we used the list of such equations, named $surf$. This list is obtained from *cond* by replacing each inequality with the equality. This is done by the following code:

```
transf={GreaterEqual->Equal,LessEqual->Equal,Greater->Equal,
                Less->Equal};
surf=cond/.transf;
```

Bounds *xmin, xmax, ymin, ymax, zmin, zmax* can be obtained using built-in MATHEMATICA functions NMinimize and NMaximize. These function provides the solution of general linear or non-linear optimization problem. Syntax for NMinimize is given by (see [9, 10]):

```
NMinimize[{f,cons},{x,y,...}]
```

Here f is expression representing a function which should be minimized, *cons* is the list of constraints and $\{x, y, ...\}$ is list of variables. Syntax for `NMaximize` is the same. Here we plug x, y and z respectively as the function f, list of constraints in (2.2) (parameter *cond* of `Geom3D`) as well as *cons* and finally $\{x, y, z\}$ as the list of variables. Calculation of *xmax*, for example, is done by the following code:

```
xmax=(NMaximize@@{{x,cond},{x,y,z}})[[1]];
```

Finally, let us notice that we used the option `RegionFunction` for plugging the list of constraints from (2.2) into the consideration. This option is set by following code:

```
RegionFunction->Function@@{{x,y,z},region}
```

Expression *region* is obtained from list *cond* by weakening each inequality by the small value *eps*. This is done in order to avoid numerical errors which may occur in `ContourPlot3D`. This is accomplished by the following code:

```
eps=Max[xmax,ymax,zmax]/1000;
transf1={GreaterEqual[a_,b_]->GreaterEqual[a,b-eps],
LessEqual[a_,b_]->LessEqual[a,b+eps],
Greater[a_,b_]->GreaterEqual[a,b-eps],
Less[a_,b_]->LessEqual[a,b+eps]};
region=And@@(cond/.transf1);
```

Step 2. Visualization of the set of extreme points.

Let us recall that for a given convex set $A \subseteq \mathbb{R}^3$, point $x \in A$ is called extremal point if there are no points $x_1, x_2 \in A$ as well as a real number $\lambda \in (0, 1)$ such that $x = \lambda x_1 + (1 - \lambda)x_2$. Since the feasible set Ω_P is convex polytope, extremal points of Ω_P are its nodes. Goal function $f(x, y, z)$ reaches its optimal value in one of these nodes. Each node is obtained by the intersection of (at least) three sides of Ω_P. It is given by the unique solution of three equations of the form (2.3). We generate and solve every such system of three equations of the form (2.3). Solutions satisfying other constraints from (2.2) are required extremal points. The extremal points are generated by the following code:

```
allpts=Subsets[surf,{3}];
nodes={};
Do[
  repl=Solve[allpts[[i]],{x,y,z}];
  If [Length[Flatten[repl]]->3,
   If[And@@(cond/.repl[[1]]),
     AppendTo[nodes,{x,y,z}/.repl[[1]]]
     ];
   ];
  ,{i,Length[allpts]}
  ];
```

List *allpts* is the list of all three equation systems obtained from equations of the form (2.3). Note that the list of these equations is already computed as the list *surf*. Built-in MATHEMATICA function Subsets produces the list of all k-elements subsets of the given set A (its parameters are k and A, given as list). We solve every system and check if it has unique solution. Furthermore, if this solution satisfy conditions *cond*, it is appended to the list *nodes* of all extremal points. After computed, these extreme points are visualized using the following expression

```
ptsplot=Graphics3D[Table[{PointSize[0.02],Point[nodes[[i]]]},
                         {i,Length[nodes]}]];
```

Step 3. Interactive visualization of the goal function and the constraint set.

Finally we visualize the objective function and provide to the user possibility to change its value continually. Such possibility is fundamental in understanding the geometrical method of linear programming. Interactivity is obtained by built-in MATHEMATICA function Manipulate. Hence user has the ability to change value d of the cost function (using the slider) and see the intersection between set of feasible solutions Ω_P and the plane $H_{f,d} = \{(x, y, z)|c_1x + c_2y + c_3z = d\}$ which contains the points where goal function has value d. This is done by the following code:

```
Manipulate[
  line=ContourPlot3D@@{c->a,{x,0,2*xmax},{y,0,2*ymax},{z,0,2*zmax},
  Mesh->None};

  Show[ptsplot,feas,line,Boxed->False,
       AxesEdge->{{-1,-1},{-1,-1},{-1,-1}},Axes->True,
       PlotRange->{{0,2*xmax},{0,2*ymax},{0,2*zmax}},
       LabelStyle->{FontSize->19},AxesStyle->Thickness[0.004]]
    ,{{a,fmin,"Goal function"},fmin,fmax},
    SaveDefinitions->True
]
```

Example 2. Consider the following example:

```
Geom3D[x+y+z,{x+y+z->1,2x+y->1,x+y->1,2y+z->4/3,z+x->2/3,
       x->0,y->0,z->0,y->0.6,z->0.5},{x,y,z}]
```

By applying the previous code we obtain the nice interactive demonstration. One snapshot is given on the figure 2. Demonstration also provides ability to the user to rotate image and see from different angles and points of view.

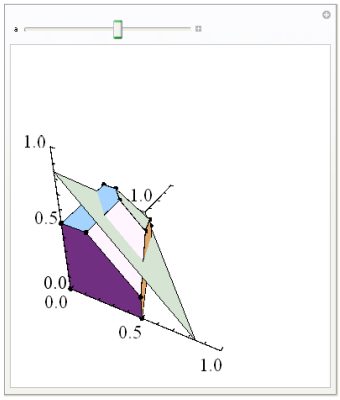

Figure 2. Snapshot of the interactive demonstration applied on the Example 2.

On figure 3 we showed the position of the goal function plane $H_{f,d}$ and feasible solutions set Ω_P, from two different points of view, in the case when the optimal solution is produced. It can be clearly viewed the set of all optimal points (in this case, it is a convex quadrilateral), and also concluded that this set is convex.

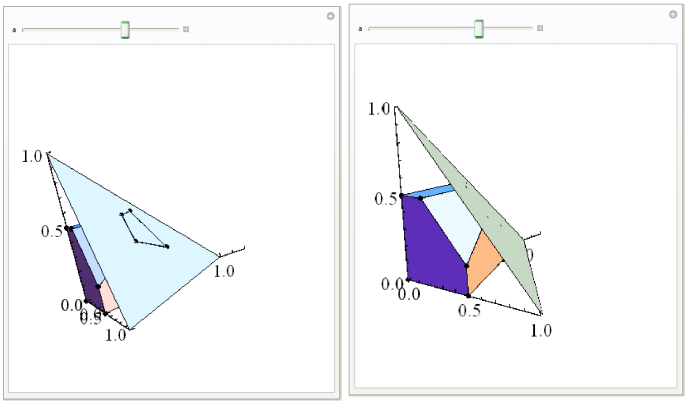

Figure 3. Two positions of the goal function plane when it obtains the maximal value.

3 Conclusion

The main aim of the present paper is to develop a software for graphical representation of the linear programming model and animation of geometrical method. This software is useful in understanding the geometrical method of linear programming. Our goal is mainly achieved using graphical primitives in `MATHEMATICA` in both two-dimensional and three-dimensional graphics. In addition, we use the symbolic possibilities of `MATHEMATICA`'s computer algebra system as well as its power in the functional and rule-based programming.

Acknowledgement: Authors wishes to thank to anonymous referee for useful comments improving the quality of the paper. This work is supported by Serbian Ministry of Science, projects 144011 and 144032.

4 Apendix

```
<< Graphics`InequalityGraphics`;
<< Graphics`Animation`;
<< Algebra`InequalitySolve`

Geom2D[f_, g_List] :=
  Module[{plot, var, region, points, txt, objective, h, n, n1, n2, g1,
     i, j, t, mx, my, res = {}, res2 = {}},
   var = Variables[f];
   region = InequalityPlot[g, {var[[1]]},{var[[2]]}, AspectRatio->1,
                   DisplayFunction->Identity];
   h = g /. {List -> And}; h = InequalitySolve[h, var];

   If [h == False, Print["Problem is infeasible!!!!"]; Break[]; ];
   g1 = g /. {LessEqual -> Equal, GreaterEqual -> Equal, Less -> Equal,
                Greater -> Equal};

   For[i = 1, i <= Length[g1] - 1, i++,
    For[j = i + 1, j <= Length[g1], j++,
      t = FindInstance[g1[[i]] && g1[[j]] && h, var];
      If[t != {},
           AppendTo[res, {t[[1, 1, 2]], t[[1, 2, 2]]}];
           AppendTo[res2, {ReplaceAll[f, t[[1]]], t[[1]]}];
       ];
      ];
    ];

   res2 = Union[res2]; res = Union[res];

   points = ListPlot[res, PlotStyle -> {PointSize[0.025], Hue[1]}];
   txt = Graphics[
            Table[{Text["A" (Subscript[i]), res[[i]] + res[[1]]/7]},
                          {i,Length[res]}]
         ];
   objective = Solve[f == 0, var[[2]]][[1, 1, 2]];

   mx = Max[Table[res[[i, 1]], {i, 1, Length[res]}]];
   my = Max[Table[res[[i, 2]], {i, 1, Length[res]}]];

   plot = Show[region, points, txt,
     TextStyle -> {FontFamily -> "Times", FontSize -> 14},
     AxesLabel -> TraditionalForm /@ var,     AspectRatio -> 1,
     PlotRange -> {{-mx/10, mx*1.1}, {-my/10, my*1.1}}];

   If[res2[[Length[res2], 1]] == res2[[Length[res2] - 1, 1]],
            Print["Optimal solution is given by \[Lambda]*", res[[Length[res]]],
            "+(1-\[Lambda])*", res[[Length[res] - 1]],",
                    0\[LessEqual]\[Lambda]\[LessEqual]1"],
```

```
            Print["Optimal solution is: ", res[[Length[res]]]];
        ];

   For[i = 1, i <= Length[res], i++,
       n = ReplaceAll[var[[2]] - objective, res2[[i, 2]]];
    ];
   n1 = ReplaceAll[var[[2]] - objective, res2[[1, 2]]];
   n2 = ReplaceAll[var[[2]] - objective, res2[[Length[res2], 2]]];

   Manipulate[Show[plot,
       Plot[objective + n, {x, -mx/10, mx*1.1},
       PlotStyle -> {Thickness[0.007], RGBColor[0, 0, 1]}]
      ], {n, n1, n2}] // Return;
   ];

Geom3D[c_,cond_,{x_,y_,z_}]:=
Module[{region,transf,surf,surf1,transf1,eps,feas,line,xmax,ymax,zmax,ret,
        fmax,allpts,nodes,repl,i,ptsplot,temena},
  xmax=(NMaximize@@{{x,cond},{x,y,z}})[[1]];
  ymax=(NMaximize@@{{y,cond},{x,y,z}})[[1]];
  zmax=(NMaximize@@{{z,cond},{x,y,z}})[[1]];
  fmax=(NMaximize@@{{c,cond},{x,y,z}})[[1]];

  transf={GreaterEqual->Equal,LessEqual->Equal,Greater->Equal,Less->Equal};
  eps=0.001;
  transf1={GreaterEqual[a_,b_]->GreaterEqual[a,b-eps],
           LessEqual[a_,b_]->LessEqual[a,b+eps],
            Greater[a_,b_]->GreaterEqual[a,b-eps],
            Less[a_,b_]->LessEqual[a,b+eps]};
  surf=cond/.transf;
  region=And@@(cond/.transf1);   allpts=Subsets[cond/.transf,{3}];
  Off[Solve::"svars"];
  nodes={};
  Do[   repl=Solve[allpts[[i]],{x,y,z}];
        If [Length[Flatten[repl]]==3,
        If[And@@(cond/.repl[[1]]),
            AppendTo[nodes,{x,y,z}/.repl[[1]]]
        ];
  ];
  ,{i,Length[allpts]}
];
ptsplot=Graphics3D[Table[{PointSize[0.02],Point[nodes[[i]]]},{i,Length[nodes]}]];
On[Solve::"svars"];

feas=ContourPlot3D@@{surf,{x,0,xmax},{y,0,ymax},{z,0,zmax},
   RegionFunction->Function@@{{x,y,z},region},
   BoxRatios->{1,1,1},Mesh->None,Axes->False,PlotPoints->30};

ret=Manipulate[ line=ContourPlot3D@@{c==a,{x,0,2*xmax},{y,0,2*ymax},{z,0,2*zmax},
                Mesh->None};
Show[ptsplot,feas,line,Boxed->False,AxesEdge->{{-1,-1},{-1,-1},{-1,-1}},Axes->True,
       PlotRange->{{0,2*xmax},{0,2*ymax},{0,2*zmax}},
         LabelStyle->{FontSize->23},AxesStyle->Thickness[0.004]]
              ,{a,0,1.5*fmax},SaveDefinitions->True
]//Return;
];
```

References

[1] P. Abbott, *Teaching Mathematics Using Mathematica*, Proceedings of the 2nd Asian Technology Conference in Mathematics (1997), 24-40.

[2] M.A. Bhatti, *Practical optimization methods with Mathematica applications*, Springer-Verlag, New York, 2000.

[3] E. K.P. Chong, S.H. Zak, *An Introduction to Optimization*, John Wiley and Sons, Inc., New York, Chichester, Weinheim, Brisbane, Singapore, Toronto, 2001.

[4] Jonassen, T.M., *Mathematica as a teaching tool for a large audience of students*, International Arctic Seminar 2002, Murmansk, Russia, May 2002.

[5] C. Loehle, *Global optimization using Mathematica: A test of software tools*, Mathematica in Education and Research (2006), 139-152.

[6] H. Ohtsuk, *Computer Technology in Mathematical Reasearch and Teaching*, Third Asian Technology Conference in Mathematics, August 24-28, (1998), University of Tsakuba, Japan, Paper Presentations.

[7] M. Sakaratovitch, *Linear programming*, Springer-Verlag, New York, 1983.

[8] K. Schittkowski, *Multicriteria Optimization -User's Guide-*, http://www.klaus-schittkowski.de, November 2004.

[9] S. Wolfram, *The Mathematica Book*, 4th ed., Wolfram Media/Cambridge University Press, 1999.

[10] S. Wolfram, *Mathematica Book*, Version 5, Wolfram Media, 2003.

[11] K. Zotos, *Performance comparison of Maple and Mathematica*, Appl. Math. Comput. **188** (2007), 1426–1429.

Predrag S. Stanimirović,
University of Niš, Faculty of Science and Mathematics,
E-mail: pecko@pmf.pmf.ni.ac.yu

Marko D. Petković,
University of Niš, Faculty of Science and Mathematics,
E-mail: dexterofnis@gmail.com

Milan Lj. Zlatanović,
University of Niš, Faculty of Science and Mathematics,
E-mail: zlatmilan@yahoo.com