Chapter 5

Using MATLAB Optimization Toolbox

In this Chapter, the use of MATLAB Optimization Toolbox is presented. This program is an extremely user-friendly numerical analysis and simulation tool. It facilitates, in particular, the use of vectors and matrices, hence its name, which can be translated as Matrix Lab.

The MATLAB program installation package can include several toolboxes each dedicated to a specific application, bringing together preprogrammed algorithms from that area. Here, the optimization toolbox will be used.

The basic ideas of the program and its usual commands are NOT given in this book. To learn that, the reader should look for one of the many good books on the subject and we will consider he knows how to use the basic tools of the program.

5.1 Optimization functions of MATLAB Toolbox

The MATLAB optimization toolbox includes the following 6 basic functions.

fminbnd – Optimization of a function of a single variable within a fixed range, ie:

find $x \in [x_L x_U]$ that minimizes $f(x)$

fminunc, fminsearch – Minimization **without restriction** of several variable functions, ie:

find **x** that minimizes $f(\mathbf{x})$

fmincon – Minimization of several variable function **with constraints**, ie:

find **x** that minimizes $f(\mathbf{x})$ subjected to

$$\mathbf{Ax} \le \mathbf{b},\ \mathbf{Nx} = \mathbf{e}$$
$$g_i(\mathbf{x}) \le 0,\ i = 1,...,m$$
$$h_j(\mathbf{x}) \le 0,\ j = 1,...,p$$
$$x_{iL} \le x_i \le x_{iU}$$

linprog – Linear programming, ie:

find **x** that minimizes $f(\mathbf{x}) = \mathbf{c}^{\mathbf{T}}\mathbf{x}$ subjected to

$$\mathbf{Ax} \le \mathbf{b},\ \mathbf{Nx} = \mathbf{e}$$

quadprog – Quadratic programming, ie:

find $\mathbf{x}$ that minimizes $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$ subjected to

$\mathbf{A}\mathbf{x} \leq \mathbf{b}$, $\mathbf{N}\mathbf{x} = \mathbf{e}$

These functions issue outputs that the user must verify:

x – is the vector or solution matrix found by the optimization function used. If ExitFlag> 0, then x is the solution; if not, x is the last value obtained by the function.

FunValue – the objective function value, ObjFun, for solution x.

ExitFlag – The output condition code of the optimization function. If ExitFlag> 0, then x is the solution; if not, x is the last value obtained by the function; if null, the maximum number of evaluations of the objective function was reached; if negative, the routine did not converge to a solution.

Output – It is an output structure of the optimization function that contains information about the process.

(Output.iterations), Provides the number of evaluations of the objective function (Output.algorithm), the name of the algorithm used to solve the problem

It is strongly suggested that the problem should be formulated as a *main program* that calls a MATLAB optimization subroutine which in turn calls a subroutine that contains the objective function and constraint function, if necessary. The general form of the command is

[x, FunValue, ExitFlag, Output]=fminX('ObjFun', ..., 'ConFun', options, other parameters)

where ObjFun is the name of a subroutine, an .m file, which contains the objective function to be called by the optimization function. This subroutine may also contain the explicitly the objective function gradient, if necessary. If this is not given, MARLAB will numerically compute the gradient, via finite differences. options are parameters of the optimization function the user wants to change in relation to the pre-programmed default values. After that it is possible to pass other problem related parameters from the main program to the subroutines. In problems with constraints, a subroutine ConFun containing these constraints and their gradients is also required.

Next, we will only detail the use of two of these basic functions: fminserch for multivariable unconstrained optimization problems and fmincon for constrained multivariable optimization problems. These are the two most popular features of the MATLAB Optimization Toolbox.

5.1.1 Multivariable unconstrained optimization problems

We strongly recommend to write a main program MATLAB script where we may first furnish a set of options that may differ from the default setting of this function such as, typically,

```
options=optimset ('LargeScale','off','TolCon',1e-8,'TolX',1e-8);
```

Next, we provide lower and upper bound for the design variables, such as minimum and maximum thickness of plates etc. This may be line matrices, if we have a multivariable problem, or a scalar value otherwise. For example:

```
Lb=[MinX1 MinX2];Ub=[MaxX1 MaxX2];
```

It is also necessary to provide an initial design, a line matrix with initial values of the design variables:

```
x0=[InitX1 InitX2];
```

The fundamental feature of this main program script is the optimization function call:

```
[x,FunVal,ExitFlag,Output]=fminsearch('ObjFun',x0,options,Prob_data);
```

Here, Prob_data may be an optional set of physical values we wish to pass to the optimization function, in the same fashion it would be done in any computer code from a main program to the subroutines. Of course, although this is an elegant procedure, it is not necessary, as we can write those values in the subroutines themselves.

Any information not used, such as the absence of the options matrix, in a particular problem, must be substituted by an empty matrix [].

5.1.2 Multivariable constrained optimization problems

We strongly recommend to write a main program MATLAB script where we may first furnish a set of options that may differ from the default setting of this function such as, typically,

```
options=optimset ('LargeScale','off','TolCon',1e-8,'TolX',1e-8);
```

Next, we provide lower and upper bound for the design variables, such as minimum and maximum thickness of plates etc. This may be line matrices, if we have a multivariable problem, or a scalar value otherwise. For example:

```
Lb=[MinX1 MinX2];Ub=[MaxX1 MaxX2];
```

It is also necessary to provide an initial design, a line matrix with initial values of the design variables:

```
x0=[InitX1 InitX2];
```

The fundamental feature of this main program script is the optimization function call:

```
[x,FunVal,ExitFlag,Output]=...
  fmincon('ObjFun',x0,A,b,Aeq,beq,Lb,Ub,'ConFun',options, Prob_data )
```

Here, A is the matrix of coefficients of possible **linear inequality constraint equations**, b the right-hand side constants vector of these equations (the so called

"available resources"). Aeq is the matrix of coefficients of possible **linear equality constraint equations**, b the right-hand side constants vector of these equations.

Prob_data may be an optional set of physical values we wish to pass to the optimization function, in the same fashion it would be done in any computer code from a main program to the subroutines. Of course, although this is an elegant procedure, it is not necessary, as we can write those values in the subroutines themselves.

Any information not used, such as the absence of linear constraints, in a particular problem, must be substituted by an empty matrix [], as:

```
x,FunVal,ExitFlag,Output]=...
  fmincon('ObjFun',x0,[],[],[],[],Lb,Ub,'ConFun',options, Prob_data )
```

5.1.3 Linear Programming problems

A linear programming problem is set as

Minimize an objective function

$$f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \tag{5.1}$$

Subjected to a set of constraint equations

$$\mathbf{Ax} = \mathbf{b} \tag{5.2}$$

All *n* inequality constraint equations are must be transformed into equality equations by including unknow positive slack variables. Thus, the size of the design variables vector **x** will be the number *m* of real design variables we wish to determine in order to minimize *f* plus the number *n* of slack variables. In consequence, matrix A is n x m, the "available resources" vector **b** will be *n* x 1, and vector **c** will be (n+m) x 1, the same size of vector **x**.

We strongly recommend to write a main program MATLAB script where we may first furnish a set of options that may differ from the default setting of this function such as, typically,

```
options=optimset ('LargeScale','off','TolCon',1e-8,'TolX',1e-8);
```

Next, we may provide lower and upper bound for the design variables. In linear programing problems it is usual to have zero lower bound values and no upper bound values specification

We may also provide an initial design, a line matrix with initial values of the design variables. Again, in linear programming it is not usual to do so.

The fundamental feature of this main program script is the optimization function call:

```
[x,FunVal,ExitFlag,Output]=linprog(c,[],[],A,b,Lb,Ub,x0,options);
```

Any information not used, such as the absence of the upper bound and initial values, in a particular problem, must be substituted by an empty matrix [].

5.2 Examples of Structural Engineering

5.2.1 Nonlinear cable problem

Consider the steel cable (Young's modulus E = 21,000 KN/cm²) originally straight under prestress force N_0 = 50 KN of Fig. 5.1. At a point at a distance L_a from the left support and L_b from the right support, a horizontal force H = 100 KN and a vertical force V = 200 KN are applied. The design variables are u, the horizontal displacement, renamed x_1 (in cm), and the vertical v, renamed x_2 (in cm).

It is important to note that this is a non-linear problem, in which equilibrium can only be written in the deformed position, initially unknown.

The basic idea is that stable equilibrium corresponds to a minimum of the Total Potential Energy:

$$\Pi = U - W$$

where U is the strain energy and W is the work of external forces

$$W = Hu + Vv$$

The change in length is

$$\Delta_a = L'_a - L_a = \sqrt{v^2 + (L_a + u)^2} - L_a$$

$$\Delta_b = L'_b - L_b = \sqrt{v^2 + (L_b - u)^2} - L_b$$

The axial forces are

$$N_a = k_a \Delta_a$$

$$N_b = k_b \Delta_b$$

where

$$k_a = \frac{EA}{L_a} \qquad k_b = \frac{EA}{L_b}$$

Thus, the Strain Energy is

$$U = (N_0 + N_a/2)\Delta_a + (N_0 + N_b/2)\Delta_b$$

A steel cable with a length of 200 cm and transverse section A = 1 cm² was adopted, divided into two equal lengths.

Using Matlab's fminsearch function in the solution, for unconstrained minimization, results x_1 = 0.248 *c*m, x_2 = 20.6733 cm, and the Total Potential Energy at this minimum point is – 3 KJ. As an exercise, check the equilibrium of the central node.

The correspondent MATLAB script is provided in the Appendix.

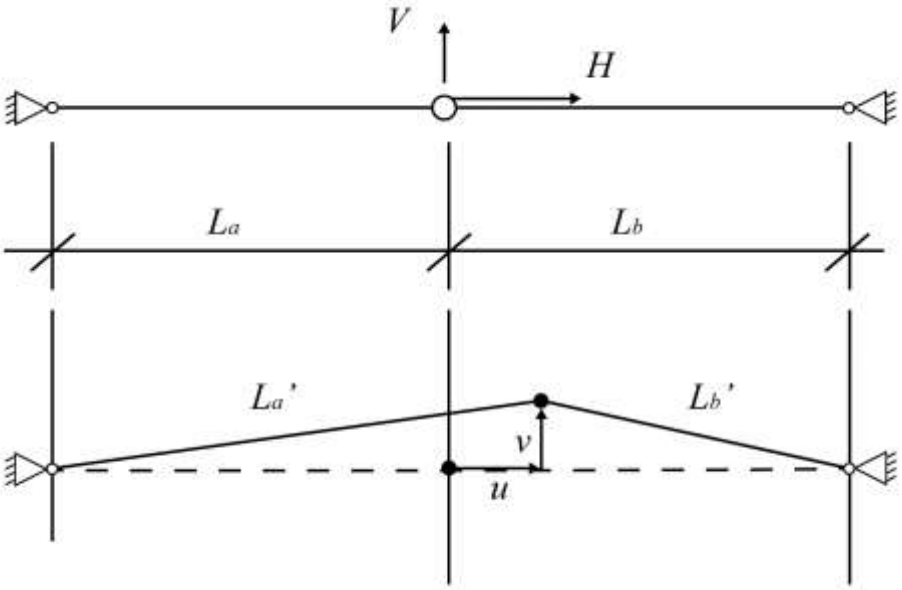

Figure 5.1

5.2.2 Eccentrically loaded tubular column

Minimize the mass of a tubular steel column of Fig. 5.2, clamped in the base and free at the upper end where an eccentric compression load is applied.

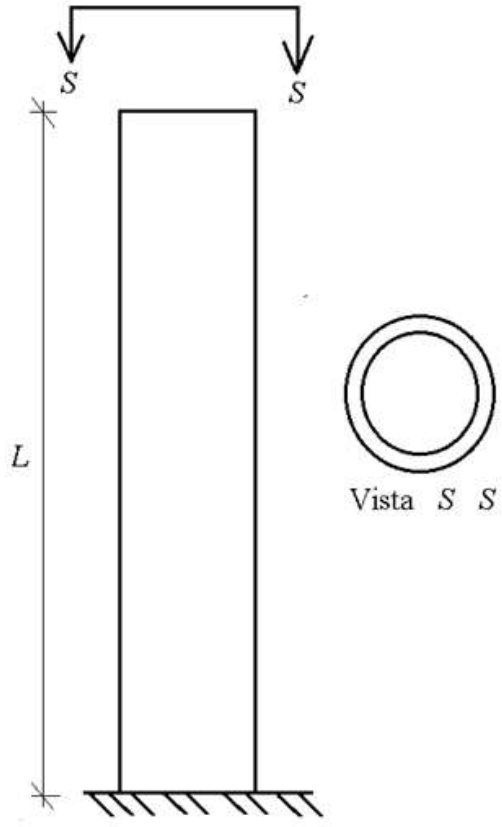

Figure 5.2 – Column to be optimized

Data:

P vertical compressive load, 100 KN

e eccentricity of load application, 2% of section average radius

L column length, 5 m

R section average radius, m

t steel wall thickness, m

E elastic modulus, 210 GPa

$\bar{\sigma}$ allowable normal stress, 250 MPa

Δ allowable displacement, 0.25 m

ρ density, 7850 kg/m³

A transverse section area, $2\pi Rt$, m²

I transverse section moment of inertia, $\pi R^3 t$, m⁴

W bending modulus, $\pi R^2 t$, m³

From the Resistance of Materials, the following design expressions are taken.

Normal stress:

$$\sigma = \frac{P}{A}\left[1 + \frac{eA}{W}\sec\left(L\sqrt{\frac{P}{EI}}\right)\right] \leq \bar{\sigma}$$

Critical buckling load:

$$P_{cr} = \frac{\pi^2 EI}{4L^2} \geq P$$

Lateral displacement:

$$\delta = e\left[\sec\left(L\sqrt{\frac{P}{EI}}\right) - 1\right] \leq \Delta$$

In addition to these constraints given by the Resistance of Materials, there are design conditions that

$$\frac{R}{t} \leq 50 \qquad \text{e} \qquad 0{,}01 \leq R \leq 1, \quad 0{,}005 \leq t \leq 0{,}2$$

The design variables are, of course, the section radius (x1) and the steel wall thickness (x2). The objective function to be minimized is the mass of the column ρLA.

Answer: R = 0.0537 m; t = 0.0050 m

The correspondent MATLAB script is provided in the Appendix.

5.2.3 Constrained optimization of a statically loaded redundant truss

Consider the 3-time redundant truss Fig. 5.3. The cross-sectional areas are x_1 (in m²) for vertical and horizontal bars, of unit length, and x_2 (in m²) for diagonal bars, of length $\sqrt{2}$ m.

It is important to remember that in a redundant (hyper-static) structure the distribution of internal forces depend on the dimensions of the cross sections.

It is assumed a gravity load P = 1 KN applied to the mobile pinned support 2. The vertical displacements p_1 and p_2 of these mobile pinned supports are the only unknowns in the solution by the Displacement Method (or stiffness method, or equilibrium method).

The properties of the material, supposed to be homogeneous, isotropic and linear elastic, are: allowable stress 10 KN/m²; elastic modulus E = 100 KN/m²; density ρ = 1000 kg/m³.

The objective function to be minimized is the total mass of the truss:

$$f = \rho \left(3\, x_1 + 2\, \sqrt{2}\, x_2\right)$$

subjected to the inequality constraint: normal stress less than the allowable stress of the material, buckling neglected.

The determination of the 2 unknown displacements, using the Displacement Method, is obtained from the algebraic linear system solution:

$$[K]\{p\} = \{P\}$$

$$E \begin{bmatrix} x_1 + x_2\ \sqrt{2}/4 & -x_1 \\ -x_1 & x_1 + x_2\ \sqrt{2}/4 \end{bmatrix} \begin{Bmatrix} p_1 \\ p_2 \end{Bmatrix} = \begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -P \end{Bmatrix}$$

The normal forces on horizontal bars 2 and 3 are zero.

The normal tensile force on vertical bar 1 is: $N_1 = Ex_1\,(p_1 - p_2)$

The normal tensile force on diagonal bar 4 is: $N_4 = -Ex_2\,p_2/2$

The normal compression force on diagonal bar 5 is: $N_5 = Ex_2\,p_1/2$

Matlab's fmincon function solution: x_1 = 0.0333 m², x_2 = 0.0943 m², minimum

total mass of the truss 366.66 kg.

The correspondent MATLAB script is provided in the Appendix.

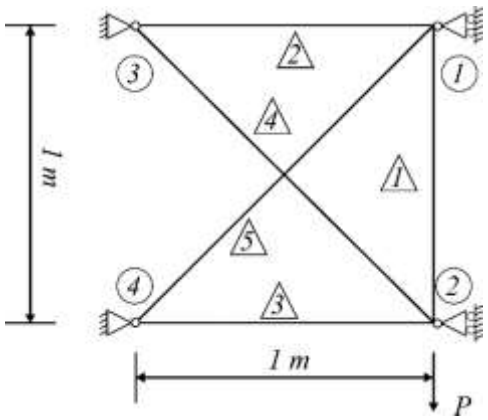

Figure 5.3

5.2.4 Frequency optimization of a redundant truss

Consider the 3-time redundant truss of Fig 5.4. The cross-sectional areas are x_1 (in m²) for vertical and horizontal bars, of unit length, and x_2 (in m²) for diagonal bars, of length $\sqrt{2}$ m. An undamped free vibration frequency bound is required.

It is important to remember that in a redundant (hyper-static) structure the distribution of internal forces depend on the dimensions of the cross sections.

The vertical displacements p_1 and p_2 of the mobile pinned supports are the only two generalized coordinates of the problem.

The properties of the material, supposed to be homogeneous, isotropic and linear elastic, are: elastic modulus E = 100 KN/m²; density ρ = 1000 kg/m³.

The objective function to be minimized is the total mass of the truss:

$$f = \rho \left(3\, x_1 + 2\,\sqrt{2}\, x_2\right)$$

subjected to the inequality constraints: the first frequency f_1 must be larger than 1 Hz and the second one f_2 less than 2 Hz. These two undamped free vibrations frequencies are obtained from the eigenvalue problem

$$\left[[K] - \omega^2[M]\right]\{\varphi\} = \{0\}$$

where the stiffness matrix is

$$[K] = E\begin{bmatrix} x_1 + x_2\ \sqrt{2}/4 & -x_1 \\ -x_1 & x_1 + x_2\ \sqrt{2}/4 \end{bmatrix}$$

and a simplified lumped mass matrix is adopted as

$$[M] = \rho\begin{bmatrix} x_1 + x_2\ \sqrt{2}/2 & 0 \\ 0 & x_1 + x_2\ \sqrt{2}/2 \end{bmatrix}$$

Matlab's function eig(*K*,*M*) returns the squares of the two frequencies, in rad/s. Matlab's fmincon function solution: x_1 =0.01m², x_2 =0.0531m², minimum total truss mass 180 kg.

The correspondent MATLAB script is provided in the Appendix.

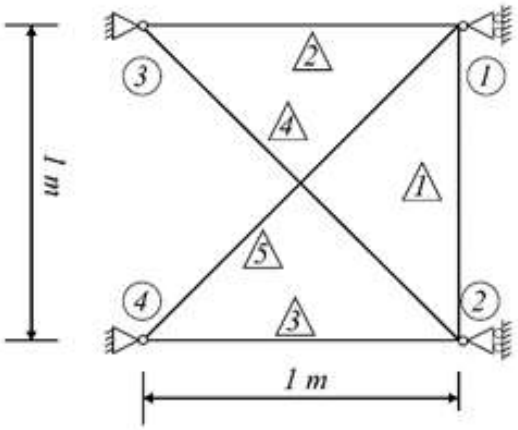

Figure 5.4

5.2.5 Thickness optimization of a rectangular steel plate simply supported under uniformly distributed loading and its own weight

Optimize the thickness *t* (with a minimum of 10 mm) of a rectangular plate simply supported under uniformly distributed loading *q* given and its own weight. The smallest dimension, *a*, is in the *x* direction and the largest dimension, *b*, in the *y* direction.

Only *b* / *a* ratios between 1 and 2 are considered, rounded to the first decimal place.

The calculations are based on Table 8 of the book “Theory of Plates and Shells”, by Timoshenko, incorporated into the Matlab program.

Steel data: *E* = 210e9 N / m², Poisson’s ratio 0.3, allowable stress 15e7 N / m², density 7850 kg / m³.

A maximum vertical displacement limit is also imposed in the middle of the span, equals to *a* / 400.

For a 4 m square plate with a load of 1 tf / m² plus its own weight, a minimum thickness of 41.5 mm was obtained using the Matlab fmincon function, corresponding to a total plate mass of 5217 kg.

The correspondent MATLAB script is provided in the Appendix.

5.2.6 Redundant wood planar portal frame

Optimize the pinned redundant wood planar portal frame of Fig. 5.5. Column height *h* = 3 m and beam span *L* = 6 m.

Data: section width b = 7.5 cm, minimum section height 15 cm, characteristic force P_k applied in the middle of the beam 10 KN, characteristic strength of wood f_{ck} = 40 MPa, average elasticity modulus E = 19,500 MPa.

As a pinned-pinned planar portal frame is once hyperstatic, the stresses in the structure depend on the sections. The horizontal reactions in the pinned supports fixed at the base of the columns, X, were chosen as hyperstatic unknowns. Their value is:

$$X = -\frac{PhL^2/8I_v}{2h^3/3I_c + Lh^2/I_v}$$

where I_v e I_c are, respectively, the section moments of inertia of the beam and the column.

Thus, the column's maximum axial force and the bending moment are

$$N_c = \frac{P}{2} \qquad M_v = Xh$$

and the beam's maximum axial force and bending moment are

$$N_v = X \qquad M_v = \frac{PL}{4} - Xh$$

(neglecting self-weight).

Verification of wood parts to design axial force N_d and bending moment M_d is carried out using a **Load and Resistance Factor Design** scheme, as follows.

Design action: $P_d = \gamma_F P_k$, where it is adopted a load factor $\gamma_F = 1.5$.

Design resistance: $f_{cd} = K_{mod}\frac{f_{ck}}{\gamma_w}$, where it is adopted resistance factor $\gamma_w = 1.4$, $K_{mod} = 0.7$ is a modification factor due to wood particular behavior.

Accidental eccentricity: $e_a = \frac{L_0}{300}$, where L_0 is the bar length.

Initial eccentricity: $e_i = \frac{M_{d1}}{N_d}$, where M_{d1} is initial design bending moment.

Design eccentricity: $e_d = (e_i + e_a)\left(\frac{F_E}{F_E - N_d}\right)$, where $F_E = \frac{\pi^2 EI}{L_0^2}$ is Euler's buckling load.

Design bending moment: $M_d = N_d e_d$.

Verification: $\left(\frac{N_d}{A} + \frac{M_d}{W}\right)/f_{cd} \leq 1$.

where A and W are the section area and the elastic bending modulus.

Using Matlab's fmincon function, we arrive at a section of 7.5x15 cm for the columns and 7.5x27.5 cm for the beam. The minimum total mass of the frame is 191.4 kg.

The correspondent MATLAB script is provided in the Appendix.

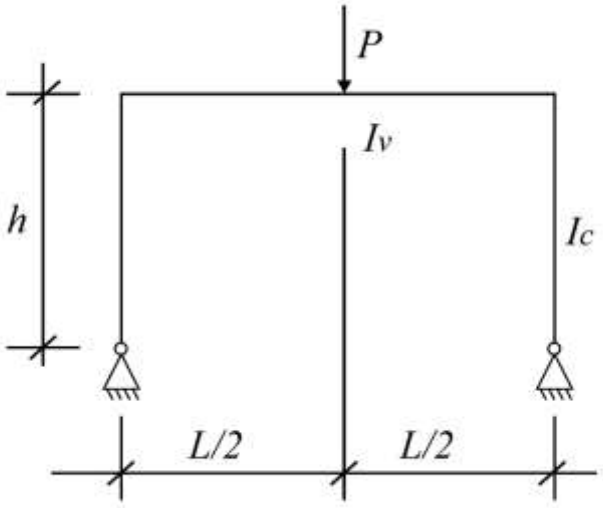

Figure 5.5

5.3 Linear Programming

We revisit example 1.4 of Chapter 1, the same as example 3.1 of Chapter 3 and example 4.1 of Chapter 4**,** of maximization of profit of a toy factory (see Chapter 4):

$$f = -400x_1 - 600x_2$$

This is an example of a problem with linear constraint equations. These are:

$$x_1 + x_2 + x_3 = 16$$

$$x_1/28 + x_2/14 + x_4 = 1$$

$$x_1/14 + x_2/24 + x_5 = 1$$

where x_1 and x_2 are, respectively, the number of type A and type B toys produced. The other 3 design variables are slack variables, that is, surplus resources. In this case, related, respectively, to delivery, production and sales capabilities.

We call linprog MATLAB function

[x,FunVal,ExitFlag,Output]=linprog(c,[],[],A,b,Lb,[],[],options)

where

$$[\mathrm{c}] = [-400 \quad -600 \quad 0 \quad 0 \quad 0]$$

$$[\mathrm{A}] = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1/28 & 1/14 & 0 & 1 & 0 \\ 1/14 & 1/24 & 0 & 0 & 1 \end{bmatrix}$$

$$[\mathrm{b}] = [16 \quad 1 \quad 1]$$

We get the same results as those of the previous methods, that is:

$$[x] = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] = [4 \quad 12 \quad 0 \quad 0 \quad 3/14]$$

x_1 and x_2 are, respectively, the number of type A and type B toys that must be produced for the maximum profit objective function value f = 8,800 Euros.

The slack variables, that is, surplus resources, indicate that delivery and production departments are fully occupied and there is a little surplus capacity in the sales department.

The correspondent MATLAB script is provided in the Appendix.