Nonlinear Optimization Models and Solving Algorithms based on Appropriate Neural Networks

Nicolae POPOVICIU
University Hyperion of Bucharest, ROMANIA
nicolae.popoviciu@yahoo.com

Abstract. This work contains a complete set of algorithms for several quadratic and nonlinear optimization problems. The problem constraints are very different. For each type of constraint an appropriate algorithm is given. The algorithms for linear bound constraints and nonlinear optimization are based on neural networks [2] [11] and uses a system of differential equations. In order to reduce the sensitivity and round off errors a preconditioning method is used. A great number of numerical applications illustrates the algorithms.

Key Words. Quadratic programming, quadratic optimization, linear bound constraints, nonlinear convex optimization, nonlinear convex bounded optimization.

1 Notations and matrix dimensions

We use the square matrices $M$ of the type $n \times n$ or rectangular matrices $M$ and denote, for example,

- $M = M_{n \times n}$ or $M \in \mathbb{R}^{n \times n}$, $M = (a_{ij})$ or
- $M = M_{m \times n}$ or $M \in \mathbb{R}^{m \times n}$, $M = (a_{ij})$.

All the used vectors are column vectors i.e. $x, c, d, p, r, u, v, w$ and denote, for example,

- $x = x_{n \times 1}$ or $x \in \mathbb{R}^n$, $x = (x_i)$.

The letter $T$ means transposition.

2 Several nonlinear optimization models

There are a lot of quadratic optimization (QO) models (or quadratic programming (QP) models) and nonlinear optimization models (NO) and here we mention several of them. We denote by $\theta$ the null vector of an appropriate space, let us say $\mathbb{R}^n$ and $x \in \mathbb{R}^n$ is the unknown vector of the any optimization problem.

Model 1. The unconstraint model.
Find $x$ so that

$$[\text{min}] F(x); F(x) = \frac{1}{2} x^T Q x - c^T x.$$ 

The input data are $n, Q, c$. The unconstrained solution is obtained from $\text{grad} F(x) = \theta$.

If $Q = Q_{n \times n}$ is a positive definite matrix, then the solution is $Q x - c = \theta, x = Q^{-1} c$.

Although the global minimum exists, for large $n$ the inversion process becomes computationally intensive and numerically unstable for ill-posed problems.

Remark 1. The matrix $Q$ could be an invertible or non invertible matrix, but always it is a symmetric matrix, because we can express

$$\frac{1}{2} x^T Q x = \frac{1}{2} x^T Q_s x, Q_s \in \mathbb{R}^{n \times n}$$

where $Q_s = \frac{1}{2} (Q + Q^T)$ (symmetric) and

$$Q_d = \frac{1}{2} (Q - Q^T)$$ (asymmetric), $x^T Q_d x = 0$ [2], [7].

Model 2. The classical QP model.
Find the vector $x$ so that

$$[\text{min}] F(x); F(x) = \frac{1}{2} x^T Q x - c^T x$$

$$C x = d, x \geq \theta$$

$$Q = \{q_{ij}\}, Q = Q_{n \times n}$$

$$C = \{c_{ij}\}, C = C_{m \times n} ; c = c_{n \times 1}$$

$$x = x_{n \times 1}, d = d_{m \times 1}.$$ 

If $Q^{-1}$ exists, then the solution $x$ is obtained by Hildreth D’Esopo algorithm [4] (which will be described in this paper).

Model 3. The QP model with bilateral linear bound constraints [2].
Find the vector $x$ so that
Proceedings of the 11th WSEAS International Conference on Sustainability in Science Engineering

[\min] F(x); F(x) = \frac{1}{2} x^T Q x - c^T x , q_{ii} > 0
\newline p \leq x \leq r , \text{ with } p_{n \times 1}, r_{n \times 1} \text{ known vectors.}

The vector inequalities are true for each corresponding components.

Model 4. The QP model with one quadratic constraint [8].

Find the vector \(x\) so that

\[ [\min] F(x); F(x) = \frac{1}{2} x^T Q x , x^T C x = 1 \]

\( C = C_{n \times n} \).

Remark 2. For the models 3 and 4 the solving algorithms are based on neural network method.


Find the vector \(x\) so that

\[ [\min] F(x), F : R^n \to R , \text{ differentiable, convex} \]

\[ Cx = d, C = C_{m \times n}, d = d_{m \times 1}, x \geq 0 \]

p \leq x \leq r.

Model 6. The nonlinear convex optimization, with bilateral linear bound constraints.[11] (The extension of model 5)

Find the vector \(x\) so that

\[ [\min] F(x), F : R^n \to R , \text{ differentiable, convex} \]

\[ Cx = d, C = C_{m \times n}, d = d_{m \times 1}, x \geq \theta \]

p \leq x \leq r.


Denote \(\Omega_2 = \{x \in R^n / Cx = d, x \geq \theta\}\). A differentiable vector function \( G : R^n \to R^n \) is given. Find the vector \(x^* \in \Omega_2\) so that

\[(x - x^*)^T G(x^*) \geq 0, \forall x \in \Omega_2\].

Now, shortly we announce that in this work our aim is to solve the model 2 (by Hildreth-D’Esopo algorithm; section 4), model 3 (by preconditioning techniques and Neural Networks; section 3), models 5,6,7 (by Neural Networks; section 5).

3 A neural network method for QP model 3 with linear bound constraints.

3.1 A special neural network architecture and the associated differential system

Our main task is to substitute the model 3 by a QP model based on a neural network (NN) architecture and to develop a solving algorithm based on this idea.

Using the consistent work [2], III, page 294 we consider the NN architecture:

1). a single-layered recurrent neural network with \(n\) processing elements (PE); the number of PE proceeds from dimension of matrix \(Q\), because

\( Q = Q_{n \times n} \) (known) and \( x \in R^n \) (unknown) ;

2). the NN has a state vector \(u \in R^n\) and this is governed by a (dynamic) differential system.

Version 1 (Use one preconditioning matrix \(B\)).

The differential system is based on \(Qx = c:\)

\[
\frac{du}{dt} = -D u - Ef(u)
\]

\( D = \text{diag}(QB) \) (main diagonal matrix) \hfill (2)

\( B = \text{diag}(d_{i1}, \ldots, d_{ii}, \ldots, d_{nn}), d_{ii} > 0\)

\( B = \text{diag}(b_{i1} \ldots b_{ii} \ldots b_{nn}), b_{ii} > 0\)

\( Qx = c \), \( x = Bf(u) \), \( QBf(u) = c \)

\( f(u) = (f_1(u_1) \ldots f_i(u_i) \ldots f_n(u_n))^T \)

\[
\frac{du}{dt} = \left( \frac{du_1}{dt} \ldots \frac{du_i}{dt} \ldots \frac{du_n}{dt} \right)^T.
\]

Version 2 (Use two preconditioning matrices \(A\) and \(B\), where \(A=B\)).

\( Qx = c, AQx = Ac \); denote \( Ac = c_0 \), \( c_0 \in R^n\).

The differential system is based on \(AQx = c_0\)

\[
\frac{du}{dt} = c_0 - Du - Ef(u)
\]

\( D = \text{diag}(AQB) \), (main diagonal matrix) \hfill (5)

\( E = AQB - D, x = Bf(u) \)

The meaning of function \(f\) remains the same as in previous version.

3), we solve the system (1) to obtain \(u\) and then obtain \(x\) from (3); analogous for version 2, solve (4) and apply (5).

Based on this ideas we will develop an algorithm to solve the model 3.

In this point, the function \(f = (f_i)\) and the matrix \(B\) (version 1) or the matrices \(A\) and \(B\) (version 2) are unknowns and these must be defined and found by the solver/user. Nevertheless, for the moment, we mention that each \(f_i\) is a sigmoid function with the slope \(s_i\).
Remark 3. We ask ourselves why is this method called a method based on neural networks. The reason is the following: here one uses the general ideas of artificial neural network, namely we use some inputs, use the answer functions $f_i$ (generally, they are sigmoid functions) and obtain a result. Actually, a neural network is an algorithm with special rules where the design is not important. The design only facilitates the problem understanding.

Remark 4. The purpose and definition of the matrix $B$ (version 1) or $A$ and $B$ (version 2) are presented in [2] and it was explained in details in [7] by algorithm 1 and algorithm 2. Here we only mention the main ideas on which was based the construction [7] of preconditioning matrices.

3.2 Condition number of a matrix and the preconditioning method

With general notations, the condition number $CN$ of a square matrix $M = (m_{ij})$ is a positive real number $CN(M)$ defined [5], [8], [6], [2] as

$$CN(M) = \|M\|_\infty \|M^{-1}\|_\infty$$

and it is estimated by

$$CN(M) \geq \frac{\max_i |m_{ii}|}{\min_i |m_{ii}|}$$

(6)

where $\lambda = \lambda(M)$ is one eigenvalue of matrix $M$.

It is known that if the matrix is real and symmetric, then all eigenvalues are real numbers. The first relation of (6) is the best evaluation of CN.

A matrix having a condition number near unity is said to be well-conditioned, whereas a large condition number indicates that the matrix is ill-conditioned.

It is known that if the matrix $M$ has a large condition number, then considerable round-off errors accumulate when attempting to solve the corresponding linear system of equations. The condition number is the error amplification factor [2].

The matrix $M$ is said to be diagonal dominant if

$$|m_{ii}| > \sum_{j=1}^{n}|m_{ij}|, \quad i = 1, m$$

(7)

The solution of a linear system of equations with the matrix $M$ non diagonal dominant is very sensitive [6], [2] just for a small changing of elements $m_{ij}$.

In our case the role of $M$ is occupied by matrix $Q$.

The preconditioning method is a technique of choosing the matrix $B$ so that the condition number of $QB$ to be very small compared with that of $Q$ i.e.

$$CN(QB) << CN(Q)$$

$B$ is called the preconditioner [2], page 299.

The susceptibility for round-off errors is considerably reduced by applying the preconditioning method.

There are several possibilities for preconditioning.

Version 0. The ideal preconditioning consists of taking the matrix $B$ as

$$B = Q^{-1}$$

because $CN(QB) = CN(I) = 1$.

Version 1. We are using the diagonal preconditioning, i.e. preconditioner $B$ is one positive diagonal matrix defined by

$$b_{ii} = \frac{k}{q_{ii}} = b_i, \quad k > 0, \quad b_{ij} = 0, \quad i \neq j$$

(8)

where $k$ is unknown so far.

Version 2 works with (1), (3) and the matrixes $D = \text{diag}(QB)$ and $E = QB - D$ (9)

Remark 5. Now we have to explain who is $k$.

The value of $k$ (version 1) is given by algorithm 1.

The value of $k$ (version 2) is given by algorithm 2.

3.3 Preconditioning algorithm 1

We denote by
\[ \lambda = \lambda(Q), \mu = \mu(QB), \rho = \rho(AQB) \]

the eigenvalues of \( Q, QB, AQB \) respectively.

Because the preconditioning method is based on eigenvalues, in numerical applications we recommend the using of MathCAD library. For example, the eigenvalues are obtained by instruction \textit{eigenvals}(Q) and the result is a column vector.

The algorithms which we intend to describe also use the solving of a polynomial equation. For example, in order to solve the equation in variable \( t \)
\[ at^3 + bt^2 + ct + d = 0 \]
we use the following steps:

Define \( v = (d, c, b, a)^T \) vector of coefficients.

Define \( r = \text{polyroots}(v) \), \( r = \cdots \) all roots. The variable identifiers \( v \) and \( r \) are at user’s disposal.

**Rule 1** (for algorithm 1). We set the condition [2]
\[ \mu(QB) = \lambda_{\text{max}}(Q) \] (12)
Step 1. Compute all eigenvalues of \( Q \) and choose the biggest one \( \lambda = \lambda_{\text{max}}(Q) \).

Step 2. Define the matrix \( B \) in concordance with (8).

Step 3. Compute the product \( QB \) (which contains the unknown \( k \)) and compute the characteristic polynomial
\[ f(\mu) = \det(QB - \mu I) \]
Step 4. Apply the rule 1 and obtain the function
\[ g(k) = \det(QB - \lambda_{\text{max}} I) \]
Step 5. Solve the polynomial equation in \( k \) (use, for example Mathcad)
\[ g(k) = 0 \] and obtain \( k_1, k_2, \ldots \)
Step 6. Take as \( k \) the \textit{minim} of all real roots \( k > 0 \).

**Remark 6.** We recommend to improve \( k_{\text{min}} \) so that
\[ g(k_{\text{min}}) < \varepsilon, \varepsilon \] a given number.

Step 7. Having \( k \), compute the elements of matrix \( B \).

Step 8 (optional). Compute all the eigenvalues \( \mu \) of \( QB \) and verify that \( \lambda_{\text{max}}(Q) = \mu_{\text{max}}(QB) \).

Then solve the differential system (1) with (9)

**Remark 7.** The value \( k = k_{\text{min}} \) assures the correspondence \( \lambda_{\text{max}}(Q) = \mu_{\text{max}}(QB) \). The value \( k = k_{\text{max}} \) assures the correspondence \( \lambda_{\text{max}}(Q) = \mu_{\text{min}}(QB) \).

Hence both \( k_{\text{max}} \) and \( k_{\text{min}} \) are good, but we recommend the version \( k = k_{\text{min}} \).

### 3.4 Preconditioning algorithm 2

**Rule 2** (for algorithm 2). We set the condition [2]
\[ \rho(AQB) = \lambda_{\text{max}}(Q) \] (13)
Step 1. Compute all eigenvalues of \( Q \) and choose
\[ \lambda = \lambda_{\text{max}}(Q) \]

Step 2. Define the matrixes \( A, B \) in concordance with (10).

Step 3. Compute the product \( AQB \) (which contains the unknown \( k \)) and compute the characteristic polynomial
\[ h(\rho) = \det(AQB - \rho I) \]

Step 4. Apply the rule 2 and obtain the function
\[ p(k) = \det(AQB - \lambda_{\text{max}} I) \]

Step 5. Solve the polynomial equation in \( k \)
\[ p(k) = 0 \]

Step 6. Take as \( k \) the \textit{minim} of all roots \( k > 0 \).

Step 7. Having \( k \), compute the elements of \( A \) and \( B \).

Step 8 (optional). Compute all the eigenvalues \( \rho \) of \( AQB \) and verify that \( \lambda_{\text{max}}(Q) = \rho_{\text{max}}(AQB) \).

Then solve the differential system (1) with (11).

The remarks 4 and 5 remain valid.

### 3.5 Numerical application for algorithm 1

**Application 1.** (3.5). We use a matrix \( Q \) from [2], page 299 and look for the matrix \( B \):
\[ Q = \begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} 39.60 & 3.96 \\ 3.96 & 0.53 \end{pmatrix} \]

The matrix is symmetric but not diagonal dominant.

Eigenvalues:
\[ \lambda_1(Q) = 39.997, \lambda_2(Q) = 0.133. \]

Estimation of condition number:
3.6 MathCAD library and solving the differential system of equations

We have to solve the differential systems (1) or (4) and use a MathCAD library.

The algorithms which we intend to describe in this work are based on several computations for
1. eigenvalues and eigenvectors;
2. solving a polynomial equation;
3. solving a first order differential system.

We use the specific MathCAD subroutines.

For example, the eigenvalues of square matrix \( Q \) are obtained by instruction \( eigenvals(Q) \) and the result is a column vector.

In order to solve the real polynomial equation in variable \( t \), let us say \( at^3 + bt^2 + ct + d = 0 \) we use the following steps:

1. Define \( vcoef = (d \ c \ b \ a)^T \) the vector of coefficients (beginning with \( t^0 \));
2. Define \( ro = \text{polyroots}(vcoef) \), \( ro = \cdots \) all roots in increasing order.

The variable identifiers \( vcoef \) and \( ro \) are at user’s disposal.

For a differential system of equations we have in mind systems (1) and (4) with unknown vector \( u \) and any dimension \( n \). Let \( n=3 \) be. The subroutine is based on Runge-Kutta method, with versions \( rkfixed \) or \( Rkadapt \). The Runge-Kutta method gives a numerical solution. We describe the first version and display several steps.

1. Choose an arbitrary vector \( u \) of initial values. The first component is \( u_0 \), i.e.
   \[
   \begin{pmatrix}
   u_0 \\
   u_1 \\
   u_2
   \end{pmatrix}
   \]

2. Introduce the column vector of derivative values at any solution point \( (t,u) \), let us say \( SQP(t,u) \), which describes the right hand side of system (1) or (4). The letter \( t \) is the independent variable.

3. Use the MathCAD instruction
   \[
   SQP = rkfixed(u,t0,t1,N,QP)
   \]
   or
   \[
   SQP = Rkadapt(u,t0,t1,N,QP),
   \]
   where
   - \( u \) is the vector of initial function values,
   - \( t0 \) is the initial value of independent variable,
   - \( t1 \) is the terminal value of independent variable,
   - \( N \) is the number of solution values on \([t0,t1]\),
   - \( SQP \) identifier is at user’s disposal.

The elements \( u_0,t0,t1,N \) are parameters at user’s disposal.

For example, a sample of parameters is
\[
\begin{pmatrix}
0 \\
0 \\
0
\end{pmatrix}
\]
\[
\text{The step of variation is } \Delta = \frac{t1}{N} = \frac{40}{80} = 0.5.
\]

The instruction \( SQP = \cdots \) gives a numerical table of solutions in each point \( t \). If on this table
appears a stability (repetition) of numerical values, then just we have obtained the system solution \( u^* \) for (1) or (4). Otherwise, we have to try again by using another combination of parameters.

Both versions \( rkfixed \) and \( Rkadapt \) have the same efficiency.

Remark 8. By X-Y Plot MathCAD subroutine we can draw the graphic of each component of \( u \). So we (are obliged to) denote

\[
t = SQP^{<0>}, \quad u0 = SQP^{<1>}, \quad u1 = SQP^{<2>},
\]

\[
u2 = SQP^{<3>}.
\]

Each pair \( (t,uk) \) yields one graphic. If we put all the graphics together, we can see the convergence of system solution, if any.

**Proposition 1.** The differential system (1) with \( f(u) = u \) and \( B = I \) solves the QP unconstrained model.

**Proof.** The system (1) becomes \( \frac{du}{dt} = c - Qu \) and \( x = u, \; u = Q^{-1}c \) (if \( Q^{-1} \) exists). (END).

### 3.7 The rule of preliminary investigations on quadratic matrix and neural network algorithm for QP model 3 with linear bound constraints

Now we deals with the finding of unknown vector \( x = (x_1 \; x_2 \cdots x_n)^T \) from model 2. The linear constraints have the form \( p \leq x \leq r \) i.e.

\[
\begin{pmatrix}
p_1 \\
\vdots \\
p_n
\end{pmatrix} \leq \begin{pmatrix}
x_1 \\
\vdots \\
x_n
\end{pmatrix} \leq \begin{pmatrix}
r_1 \\
\vdots \\
r_n
\end{pmatrix}
\]

Shortly, we denote the neural network algorithm for quadratic problem (optimization) with linear constraints by NNQPLBCAlgo (or QP algorithm). This algorithm uses as output function in (1) or (4) some sigmoid functions

\[
f_i(u_i), \; i = 1, n \text{ or } i = 0, n - 1
\]

having the general form:

\[
f_i : R \rightarrow (b_i, a_i), \quad f_i(u_i) = a_i e^{s_i u_i} + b_i, \quad e^{s_i u_i} + 1, \quad a_i \in R, b_i \in R, s_i > 0 \; (s_i \text{ is the slope}).
\]

If the slope value \( s_i \) is \( s_i \geq 5 \), then the main part of sigmoid function is almost a straight line.

After we have obtained the numerical solution \( u^* \), then we define the function

\[
f_i(u_i) = \begin{cases} 
    b_i, & u_i < b_i \\
    a_i, & u_i > a_i 
\end{cases}
\]

(15)

The MathCAD uses the numbering \( i = 0, n - 1 \).

But first of all we recall several ideas about the condition number \( CN \) (or spectral condition number) of a square matrix. This is the error amplification factor. A matrix having a \( CN \) near unity is said to be well-conditioned, whereas a large \( CN \) indicates that the matrix is ill-conditioned. A method to evaluate/approximate (not to compute) the \( CN \) is

\[
CN(Q) \geq \frac{\lambda_{\max}}{\lambda_{\min}} \text{ or } CN(Q) \approx \frac{\max_i q_{ii}}{\min_i q_{ii}}.
\]

The steps of NNQPLBCAlgo are based on MathCAD procedures.

Step 1. Introduce the input data: \( Q, c, p, r \). Always \( Q \) is a symmetric matrix.

Step 2. Preliminary investigations on matrix \( Q \).

- a) If \( Q \) is diagonal dominant and \( CN \) near 1, then use the preconditioning method with \( B = I \).
- b) If \( Q \) is not diagonal dominant and \( CN \gg 1 \), then use the preconditioning method with \( B = k I \), with \( k = 2, 3, 4 \).
- c) If \( Q \) is not diagonal dominant and \( CN \gg 1 \), then use the preconditioning matrices \( A \) and \( B \), which are based on value of \( k \). For the value of \( k \) see [2] and [7].

Step 3. Find the matrix \( B \) or matrices \( A \) and \( B \).

Step 4. Investigate the determinant \( \det Q \).

- a) If \( \det Q \neq 0 \), then \( Q \) inverse exists and we can compute unrestricted minimum \( x = Q^{-1}c \).
- b) If \( \det Q = 0 \), then compute eigenvalues and eigenvectors of matrix \( Q \), for example by \( \text{eigvals}(Q) \) and \( \text{eigvec}(Q, V) \).

Sure \( \lambda_0 = 0 \) and compute the associated eigenvector \( v(\lambda_0) \).

b1. If the vector \( c \) is not orthogonal on \( v(\lambda_0) \), then we have a single constrained minimum.

b2. If the vector \( c \) is orthogonal on \( v(\lambda_0) \), then there are an infinite number of constrained minima.

Step 5. Compute the new borders

\[
p \leq x \leq r, \quad v = B^{-1}p, \quad w = B^{-1}r
\]

Denote \( v = (b_i), \quad w = (a_i), \quad i = 0, n - 1 \).
Step 6. Use (6) and define the sigmoid functions $f_i$ (the output of processing element $i$ of neural network).

Step 7. Compute the matrices $D$ and $E$ from (1) or (4).

Step 8. Denote $u = (u_0 \cdots u_{i-1} \cdots u_{n-1})^T$ and

$$f(u) = (f_0(u_0) \cdots f_i(u_i) \cdots f_{n-1}(u_{n-1}))^T.$$  

Compute the column vector of derivatives from (1) or (4), respectively

$$QP(t, u) = c - Du - Ef(u) \quad (17)$$

Step 9. Choose an initial vector $u$ and slopes $s_i$ and apply the MathCAD instruction $SQP$ with appropriate parameters

$$SQP = rkfixed(u, t_0, t_1, N, QP) \quad (18)$$

Step 10. Analyze the results of table $SQP$.

a) If the string of numerical values of $u_i$ is convergent, then select the differential system solution $u^*$ and GO TO Step 11.

b) If the string is not convergent, then change one or more parameters $u_i, s_i, t_0, t_1, N \quad (19)$

and GO TO (18).

Step 11. Apply the definition (15) and construct the function $f(u^*)$.

Step 12. Compute the optimal solution (equilibrium point) $x^* = Bf(u^*)$ and the minimum value $F(x^*)$. END.

3.8 QP model 3 with linear bound constraints applications

We use NNQPLBCA1go to solve some quadratic optimization problems. Some examples are taken from [2] and we compare author’s results with our results. In our solving are given all the computations for model 2. We denote the input matrices by $Q1, Q2, Q3 \ldots$ and $c1, c2, c3 \ldots$ etc.

In order to validate the above QP algorithm we solve the problem proposed by Bouzerdoum and Pattison [2].

**Application 1.** (3.8) The input data are

$$Q1 = \begin{bmatrix} 2 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 2 \end{bmatrix}, \quad c1 = \begin{bmatrix} 1 \\ -1 \\ 0.5 \end{bmatrix},$$

$$-1 \leq x_i \leq 1, \quad i = 1, 3 \quad (20)$$

**Solution.** The matrix $Q1$ is diagonal dominant, \( \det Q1 = 6.75 \) and $Q1^{-1}$ exists. The condition number is $CN(Q1) \approx 1$ and hence one uses a preconditioning matrix in simplest form.

**Solution 1.** Compute the unconstrained optimum

$$x1^* = Q1^{-1} c1 = \begin{bmatrix} 0.611 \\ -0.722 \\ 0.278 \end{bmatrix}.$$

Fortuitous vector $x1^*$ is solution for constrained model 3, because $x1^*$ satisfies conditions (20) and the minimum is $F(x1^*) = -0.736$.

**Solution 2.** Use the QP algorithm with one preconditioning matrix $B = I$ (proposed in [2]) and three (different) sigmoid functions $f_i, i = 0, 2$ (this is MathCAD numbering).

Compute the arguments of sigmoid functions (14):

$$p_i = -1, \quad r_i = 1, \quad i = 0, 2$$

$$v = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}, \quad B^{-1} p = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$w = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \quad B^{-1} r = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Construct the matrices $D$ and $E$ of system (1).

Take a set of slopes $s_i$ of (6) and then prepare the column vector of derivatives $Q1P(t, u)$. We obtain

$$D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad u = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix}, \quad f(u) = \begin{bmatrix} f_0(u_0) \\ f_1(u_1) \\ f_2(u_2) \end{bmatrix}$$

$$E = \begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.5 & 0.0 & 0.5 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

$$u = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (\text{initial vector})$$

$$Q1P(t, u) = \begin{bmatrix} 1 - 2u_0 - 0.5[f_1(u_1) + f_2(u_2)] \\ -1 - 2u_1 - 0.5[f_0(u_0) + f_2(u_2)] \\ 0.5 - 2u_2 - 0.5[f_0(u_0) + f_1(u_1)] \end{bmatrix}$$

Try one set of slopes for sigmoid functions and use the MathCAD instruction

$$SQ1P = rkfixed(u, 0, 2, 0, 50, Q1P).$$
Analyze the results and the solution convergence. If the result is unsatisfactory, then try another set of slopes or another initial vector \( u \) or another parameters of rkfixed subroutine.

In our example, by using the above \( u \) and parameters, we stopped the choosing of slopes with the values \( s_0 = 2.5, s_1 = 2.5, s_2 = 2 \).

The rkfixed subroutine gives a string of results for \( u_0, u_1, u_2 \), having the form

\[
\begin{align*}
0.615 & -0.720 & 0.267 \\
0.616 & -0.722 & 0.267 \\
0.617 & -0.723 & 0.267 \\
0.617 & -0.724 & 0.268 \\
0.617 & -0.724 & 0.268 \\
\end{align*}
\]

The repetition of values shows that the vector \( u_1^* \) is the solution of differential system (1).

Use the values of \( a_i, b_i \) from (21), (22), apply the definition (7) and obtain

\[
f(u_1^*) = (0.617 -0.724 \ 0.268)^T.
\]

Compute the solution \( x_1^* \) and \( F(x_1^*) \)

\[
x_1^* = B f(u_1^*), \quad x_1^* = (0.617 -0.724 \ 0.268)^T
\]

\[
F(x_1^*) = -0.736.
\]

Directly, without computations, the authors [2] give the solution

\[
x^* a = (0.6111 -0.7222 \ 0.2778)^T
\]

and we see that \( F(x^*) = -0.736 \).

**Application 2.** (3.8) The input data are

\[
Q2 = \begin{bmatrix}
2 & 1.25 & 1.25 \\
1.25 & 2 & 1.25 \\
\end{bmatrix}, \quad c2 = \begin{bmatrix}
1 \\
-1 \\
0.5 \\
\end{bmatrix}
\]

\[
-1 \leq x_i \leq 1, \ i=1,3
\]

**Solution.**

We try the unconstrained optimum and obtain

\[
\det Q2 = 2.531, \ x = Q2^{-1} c2 = \begin{bmatrix}
1.148 \\
-1.519 \\
0.481 \\
\end{bmatrix}
\]

The vector \( x \) is not satisfying the conditions (23). If we use the system (1) and the proposition 1, then one obtains \( x = (1.142 \ -1.51 \ 0.479)^T \).

The concordance of results is obviously.

**Solution 1.** The matrix \( Q2 \) is not diagonal dominant, but the condition number is small. Then a preconditioning matrix \( B = I \) is necessary.

We use the system (1) and \( B=I \) [2].

Compute the arguments of sigmoid functions (14) and choose an initial vector \( u \) as in application 1.

Compute matrices \( D, E \) and the column vector of derivatives \( Q2P1(t,u) \)

\[
Q2P1(t,u) = c2 - Du - Ef(u).
\]

Try one set of slopes for sigmoid functions and use the MathCAD instruction

\[
SQ2P1 = rkfixed(u,0.20,40,Q2P1).
\]

The set of slopes \( s_0 = 4,s_1 = 6,s_2 = 2 \) give the results

\[
\begin{array}{ccc}
0.954 & -1.265 & 0.277 \\
0.955 & -1.266 & 0.276 \\
0.956 & -1.267 & 0.276 \\
0.956 & -1.267 & 0.276 \\
0.956 & -1.267 & 0.276 \\
\end{array}
\]

and so on the some values.

The vector \( u_2^* \) is the solution of system (1).

Use the values of \( a_i, b_i \) from (21), (22), apply the definition (15) and obtain

\[
f(u_2^*) = (0.956 \ -1 \ 0.276)^T.
\]

Compute the solution \( x_2^* \) and \( F(x_2^*) \)

\[
x_2^* = B f(u_2^*), \quad x_2^* = (0.956 \ -1 \ 0.276)^T
\]

\[
F(x_2^*) = -1.314.
\]

The author’s solution [2] is

\[
x_2^* = (0.9487 \ -1 \ 0.2821)^T
\]

\[
F(x_2^*) = -1.314.
\]

**Remark 9.** System (1) and different settings of slopes give the following results

\[
\begin{array}{ccc}
s_0 & s_1 & s_2 \\
4 & 6 & 2 \\
5 & 6 & 2 \\
5 & 5 & 2 \\
6 & 5 & 2 \\
5 & 5 & 3 \\
\end{array}
\]

\[
\begin{array}{ccc}
f(u^*) & F(x^*) \\
0.956 & -1.267 & 0.276 & -1.314 \\
0.966 & -1.274 & 0.259 & -1.314 \\
0.965 & -1.273 & 0.258 & -1.314 \\
0.969 & -1.275 & 0.252 & -1.314 \\
0.889 & -1.345 & 0.263 & -1.309 \\
\end{array}
\]
This table shows the sensibility of QP algorithm, namely the relation between parameters and the results

Solution 2. In order to accelerate the algorithm convergence we take $B = 4I$ (author’s proposal). In this case very quickly we obtain the stabilization of solution as it follows:

\[
B^{-1} = \begin{pmatrix} -1 & -0.25 \\ -1 & -0.25 \end{pmatrix}, \quad B^{-1} = \begin{pmatrix} 1 & 0.25 \\ 1 & 0.25 \end{pmatrix}
\]

\[
Q2P2(t,u) = c2 - D u - Ef(u)
\]

\[
SQ2P2 = rkfixed(u,0.20,40,Q2P2)
\]

.........................

\[
\begin{pmatrix} 0.234 & -0.259 & 0.070 \\ 0.237 & -0.263 & 0.071 \\ 0.238 & -0.264 & 0.071 \\ 0.238 & -0.264 & 0.071 \\ 0.238 & -0.264 & 0.071 ; -0.264 < -0.25.
\end{pmatrix}
\]

\[
f(u2*) = (0.238 -0.25 0.071)^T
\]

\[
x2* = B(u2*) = (0.952 -1 0.284)^T
\]

\[
F(x2*) = -1.314.
\]

Application 3/(3.8) [2] Consider the following symmetric bound constrained quadratic optimization problem with the input data

\[
Q3 = \begin{pmatrix} 39.60 & 3.96 \\ 3.96 & 0.53 \end{pmatrix}, \quad c3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}
\]

\[-15 \leq x_i \leq 15, \quad i=1,3.
\]

Solution. Matrix $Q3$ is not diagonal dominant and the evaluation of condition number gives $CN(Q3) \geq 39.60/0.53 \geq 74.7$. The matrix is ill-conditioned. By system (1) the solution is $x^* = (0 \ 0)^T$, $F(x^*)=0$.

Application 4/(3.8) Consider the input data [2]

\[
Q4 = Q3, \quad c4 = (-574.20 \ -55.40)^T
\]

\[-15 \leq x_i \leq 15, \quad i=1,3.
\]

Solution. The comments from application 3, related with $Q3$, show the necessity of preconditioning procedure. We use [2] and version 2 with differential system (4). The matrices $A$ and $B$ are

\[
A = B = \begin{pmatrix} k/\sqrt{39.60} & 0 \\ 0 & k/\sqrt{0.53} \end{pmatrix}, \quad k = 4.633
\]

\[
A = B = \begin{pmatrix} 0.736 & 0 \\ 0 & 6.364 \end{pmatrix}, \quad c41 = A \cdot c4
\]

\[
B^{-1} = \begin{pmatrix} 1.358 & 0 \\ 0 & 0.157 \end{pmatrix}, \quad c41 = \begin{pmatrix} -422.744 \\ -352.561 \end{pmatrix}
\]

We mention that the value of $k$ is given in [2] and is justified and proofed in [7]. Compute the matrices $AQB$, $D$, $E$

\[
AQB = \begin{pmatrix} 21.465 & 18.554 \\ 18.554 & 21.465 \end{pmatrix}
\]

\[
D = \begin{pmatrix} 21.465 & 0 \\ 0 & 21.465 \end{pmatrix}, \quad E = \begin{pmatrix} 0 & 18.554 \\ 18.554 & 0 \end{pmatrix}
\]

Compute the arguments of sigmoid function (6)

\[
v = B^{-1} \begin{pmatrix} -15 \\ -15 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} -20.374 \\ -2.357 \end{pmatrix}
\]

\[
w = B^{-1} \begin{pmatrix} 15 \\ 15 \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 20.374 \\ 2.357 \end{pmatrix}
\]

Set an initial vector $u = (u_0, u^*)^T$, the slope values and construct the sigmoid functions

\[
u = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad s_0 = 5, \quad s_1 = 5, \quad f(u) = \begin{pmatrix} f_0(u_0) \\ f_1(u_1) \end{pmatrix}
\]

\[
f_0(u_0) = \frac{a_0 e^{s_0 u_0} + b_0}{e^{s_0 u_0} + 1}
\]

\[
f_1(u_1) = \frac{a_1 e^{s_1 u_1} + b_1}{e^{s_1 u_1} + 1}
\]

Construct the matrix of derivatives

\[
Q4P(t,u) = c41 - D u - Ef(u)
\]

Use the MathCAD instruction

\[
SQ4P = rkfixed(u,0.20,400,Q4P)
\]

This instruction yields the string of values

\[
\begin{array}{ll}
-21.714 & 1.185 \\
-21.719 & 1.186 \\
-21.721 & 1.186 \\
-21.721 & 1.186 \\
\end{array}
\]

\[
(7) \ implies \ f(u^*) = \begin{pmatrix} -20.374 \\ 1.186 \end{pmatrix}
\]

Compute the optimal solution and the corresponding minimum value
\[ x^4 = Bf(u^4^\text{r}) = \begin{pmatrix} -15 \\ 7.548 \end{pmatrix} \]

\[ F(x^4^\text{r}) = -4173. \]

The authors directly give the solution \( xa \)

\[ xa = \begin{pmatrix} -15 \\ 7.538 \end{pmatrix}, \quad F(xa) = -4173. \]

**Application 5.(3.8)** Consider the constrained minimization problem with the input data

\[ Q5 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}, \quad c5 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \]

\[ -1 \leq x_i \leq 1, \quad i = 1, 3. \]

**Solution.** The characteristics of matrix are: \( \det Q5 = 0 \) (positive semi-definite), not diagonal dominant and the condition number is 1.

Because \( Q5 \) is a singular matrix we must to study the matrix eigenvalues and eigenvectors. The MathCAD subroutine gives the eigenvalues and the corresponding eigenvectors (see table with \( V \))

\[ \lambda_0 = 0 \quad \lambda_1 = 1.268 \quad \lambda_2 = 4.732 \]

\[ V: \begin{pmatrix} 0.707 \\ -0.3251 \\ 0.6280 \\ 0.6280 \\ -0.3251 \\ 0.6280 \\ 0 \\ 0.8881 \end{pmatrix} \]

The first vector of \( V \) (for \( \lambda_0 = 0 \)) is not orthogonal to the input vector \( c5 \). Then the minimization problem has a single constrained minimum.

Use the preconditioning matrix and set \( B = 2I \) ([2]). So we obtain

\[
\begin{bmatrix}
4 & 0 & 0 \\
0 & 4 & 2 \\
0 & 0 & 2
\end{bmatrix}
= \begin{bmatrix}
042 \\
220 \\
402
\end{bmatrix}
\]

\[ f(u) = \begin{pmatrix} f_0(u_0) \\ f_1(u_1) \\ f_2(u_2) \end{pmatrix} \]

The margins for functions (6) are

\[
\begin{pmatrix}
1 \\
-1 \\
-1
\end{pmatrix}
= \begin{pmatrix}
-0.5 \\
-0.5 \\
-0.5
\end{pmatrix}
\begin{pmatrix}
0.5 \\
0.5 \\
0.5
\end{pmatrix}
\]

\[
\begin{pmatrix}
b_0 \\
b_1 \\
b_2
\end{pmatrix}
= \begin{pmatrix}
1 \\
1 \\
1
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
a_2
\end{pmatrix}
\]

Construct the sigmoid functions \( f_i(u_i) \) and the vector of derivatives

\[ Q5P(t, u) = c5 - Du - Ef(u). \]

Take an initial vector \( u \) and a set of slopes, for example \( u = (0.75 \quad -0.75 \quad 0.0)^T \) and \( s_0 \in [6;10], s_1 \in [6;10], s_2 \in [6;10] \).

Use the instruction

\[ SQ5P = rknxfixed(u,0,40,200,Q5P) \]

The solution is very quickly convergent to \( u5^* = (0.623 \quad -0.623 \quad 0.125)^T \) and

\[ f(u5^*) = (0.5 \quad -0.5 \quad 0.125)^T \]

\[ x5^* = Bf(u5^*) = (1 \quad -1 \quad 0.25)^T \]

\[ F(x5^*) = -1.063. \]

In [2] the authors have the same results.

If we use the initial vector \( u = (1-10)^T \), then

\[ u5^* = (0.625 \quad -0.625 \quad 0.125)^T \]

and \( x5^* \) doesn’t change.

**Application 6.(3.8)** [2] The input data are

\[ Q6 = Q5, \quad c6 = (0.5 \quad 0.5 \quad -0.5)^T \]

\[ -1 \leq x_i \leq 1, \quad i = 1, 3. \]

**Solution.** The matrix \( Q6 \) has the same characteristics and the same eigenvalues and eigenvectors as the matrix \( Q5 \). There is a difference: the vector \( c6 \) is orthogonal to first column vector from table \( V \). Then, there are an infinite number of constrained minima. The solution is depending on the initial conditions.

As a preconditioning matrix \( B \) we set \( B = 2I \) [2]. Hence the matrices \( D, E \) and the margins \( v, w \) are the same as in application 5. The vector of derivatives from (1) is

\[ Q6P(t, u) = c6 - Du - Ef(u). \]

The MathCAD instruction

\[ SQ6P = rknxfixed(u,0,20,200,Q6P) \]

and initial vector \( u = (0.1 \quad 0.0 \quad -0.09)^T \) with the slopes \( s_0 = s_1 = s_2 = 8 \) yields the solution

\[ u6^* = (0.211 \quad 0.021 \quad -0.22)^T \]

\[ f(u6^*) = (0.211 \quad 0.021 \quad -0.22)^T \]

\[ x6^* = Bf(u6^*) = (0.422 \quad 0.042 \quad -0.44)^T \]

\[ F(x6^*) = -0.247. \]

In [2] the authors obtained
\[ \begin{align*}
xa_1 &= \begin{pmatrix} 0.25 \\ 0.25 \\ -0.50 \end{pmatrix}, \\
xa_2 &= \begin{pmatrix} -0.25 \\ 0.75 \\ -0.50 \end{pmatrix} \\
F(xa_1) &= -0.25, F(xa_2) = -0.25.
\end{align*} \]

4. **QP Hildreth-D’Esopo algorithm**

4.1 Special QP optimization models and their dual forms

We re-write the model 1 of section 2 in two different standard forms.

**Model 1.1 (P1)** Find the vector \( x \) so that

\[ \begin{align*}
\text{[min]} F(x); F(x) &= \frac{1}{2} x^T Q x + c^T x \\
C x &\geq b, b \geq \theta; \ x \in \mathbb{R}^n
\end{align*} \tag{24} \]

\[ Q = (q_{ij}), Q = Q_{m \times n} \text{ (symmetric matrix)} \]

\[ C = (c_{ij}), C = C_{m \times n}; c = c_{n \times 1} \]

\[ x = x_{n \times 1}, b = b_{m \times 1}. \]

**Model 1.2 (P2)** Find the vector \( x \) so that

\[ \begin{align*}
\text{[min]} F(x); F(x) &= \frac{1}{2} x^T Q x + c^T x \\
C x &\leq b, b \geq \theta; \ x \in \mathbb{R}^n
\end{align*} \tag{27} \]

**Remark 7.** The condition (25) \( C x \geq b \) is in accordance (naturally) with the minimization problem, while the condition (27) \( C x \leq b \) is not in accordance (not naturally) with the minimization problem. This remark is important for dual problem.

If \( Q^{-1} \) exists, then the optimal solution \( x^* \) is obtained by Hildreth D’Esopo algorithm [4]. This algorithm is based on the associated dual problem.

The dual problems of P1 and P2 are denoted by P1D and P2D, respectively.

**P1D.** Find the vectors \( x \) and \( u \) so that

\[ \begin{align*}
\text{max} \ G_1(x,u); G_1(x,u) &= -\frac{1}{2} x^T Q x + b^T u \tag{28} \\
x &\in \mathbb{R}^n, u \geq \theta, u = u_{m \times 1}
\end{align*} \]

\[ Q x - C^T u + c = \theta, \theta = \theta_{n \times 1} \tag{30} \]

**P2D.** Find the vectors \( x \) and \( u \) so that

\[ \begin{align*}
\text{max} \ G_2(x,u); G_2(x,u) &= -\frac{1}{2} x^T Q x - b^T u \tag{31} \\
x &\in \mathbb{R}^n, u \geq \theta, u = u_{m \times 1}
\end{align*} \]

\[ Q x + C^T u + c = \theta, \theta = \theta_{n \times 1} \tag{33} \]

**Proposition 2.** If the matrix \( Q \) is invertible, then the function \( G_1(x,u) \) could be expressed only with the variable \( u \).

**Proof.** From (30) and (28) we obtain successively \( x = Q^{-1}(C^T u - c) \tag{34} \)

\[ G_1(x,u) = -\frac{1}{2} c^T Q^{-1} c + h_1(u) \tag{35} \]

\[ h_1(u) = -\frac{1}{2} u^T C Q^{-1} C^T u + (b + C Q^{-1} c)^T u \]

Formula (35) contains only the variable \( u \) (End).

**Corollary 1.** The first term in (35) is a constant. So, the finding of vector \( u^* \) from \([\text{max}]G_1(x,u)\) is equivalent with \([\text{max}]h_1(u)\) and \([\text{max}]h_1(u) = [\text{max}]\{\text{h}_1(u)\} = -[\text{min}]H_1(u)\) where

\[ H_1(u) = \frac{1}{2} u^T C Q^{-1} C^T u - (b + C Q^{-1} c)^T u \tag{36} \]

Hence, we find \( u^* \) from \([\text{min}]H_1(u)\). Usually we set the matrix notations \( Dl = D_{m \times m}, dl = d_{m \times 1} \)

\[ Dl = C Q^{-1} C^T, dl = b + C Q^{-1} c \tag{37} \]

\[ H_1(u) = \frac{1}{2} u^T Dl u - dl^T u \tag{38} \]

Hence, \([\text{min}]H_1(u)\) produces \( u^* \) and then \( x^* = Q^{-1}(C^T u^* - c) \). \([\text{min}]F(x) = F(x^*)\) \tag{39} \]

**Proposition 3.** If the matrix \( Q \) is invertible, then the function \( G_2(x,u) \) could be expressed only with the variable \( u \).

**Proof.** From (33) and (31) we obtain successively \( x = -Q^{-1}(C^T u + c) \tag{40} \)

\[ G_2(x,u) = -\left[ \frac{1}{2} c^T Q^{-1} c + H_2(u) \right] \tag{41} \]

\[ H_2(u) = \frac{1}{2} u^T C Q^{-1} C^T u + (b + C Q^{-1} c)^T u \]

Formula (41) contains only the variable \( u \) (End).

**Corollary 2.** The first term in (41) is a constant. So, the finding of vector \( u^* \) from \([\text{max}]G_2(x,u)\) is equivalent with \([\text{max}]\{-H_2(u)\} = -[\text{min}]H_2(u)\).
Hence, we find $u^*$ from $[\min]H2(u)$. The matrix notations
$$D2 = CQ^{-1}C^T, d2 = b + CQ^{-1}c$$
produce $H2(u) = \frac{1}{2}u^TD2u + d2^Tu$ (42)

The matrix $D1=D2$ is symmetric matrix.

Now it is possible to compute
$$x^* = -Q^{-1}(C^Tu^* + c), \quad [\min]G2(x,u) = F(x^*)$$

We remark that for both problems P1 and P2 we find $u^*$ by $[\min]H1(u)$ and $[\min]H2(u)$, respectively.

Based on the minimization of functions $Hk(u)$ we construct the Hildreth (1957) and D’Esopo (1959) algorithm (HDE algorithm).

4.2 Hildreth-D’Esopo algorithm

We describe this algorithm ([4], page 255) for model 1.2 and the proposition 3. The aim is the vector $u = u^*, u = (u_1 \cdots u_i \cdots u_m)^T$. For that we compute the gradient vector
$$\text{grad}H2(u) = d2 + D2u = \theta$$ (43)

Step 1. Arrange the constraints in standard form (19) and input the numerical data: $n$, $m$, $Q$, $c$, $C$, $b$.

Step 2. Compute
$$Q^{-1}, D2 = (d_{ij}), d2 = (d_i), d_{ii} \neq 0$$

Step 3. Set an initial vector $u^0 \in R^m$, let us say $u^0 = \theta$. The upper index counts the iteration number $k=0$.

Step 4. Let us suppose we know the vector $u$ at iteration number $k$ i.e. $u^k = (u_1^k \cdots u_i^k \cdots u_m^k)^T$.

Compute (based on (43))
$$u_{i}^{k+1} = \max\{0;v_{i}^{k+1}\}, i = 1, m, \quad \text{where}$$
$$v_{i}^{k+1} = -\frac{1}{d_{ii}}(d_i + \sum_{j=1}^{i-1} d_{ij}u_{j}^{k+1} + \sum_{j=i+1}^{m} d_{ij}u_{j}^{k}), i = 1, m$$ (44)

STOP the iterations when the components of vector $u^k$ don’t change. It is proved [4] that the process is convergent. The last $u^k$ is $u^*$.

Step 5. Compute $x^*$ and $F(x^*)$.

Remark 10. We suggest the following arrangement of computations ($n=4$)
$$d_1 \quad d_{11} \quad d_{12} \quad d_{13} \quad d_{14}$$
$$d_2 \quad d_{21} \quad d_{22} \quad d_{23} \quad d_{24}$$
$$d_3 \quad d_{31} \quad d_{32} \quad d_{33} \quad d_{34}$$
$$d_4 \quad d_{41} \quad d_{42} \quad d_{43} \quad d_{44}$$

Iteration
$$k \quad u_1^k \quad u_2^k \quad u_3^k \quad u_4^k$$
$$0 \quad u_1^0 \quad u_2^0 \quad u_3^0 \quad u_4^0$$
$$1 \quad u_1^1 \quad u_2^1 \quad u_3^1 \quad u_4^1$$
$$2 \quad u_1^2 \quad u_2^2 \quad u_3^2 \quad u_4^2$$
$$3 \quad u_1^3 \quad u_2^3 \quad \bullet \quad \cdots$$

4.3 Application of HDE algorithm

Find $x$ so that
\[
[x_1 \quad x_2] = \begin{pmatrix}
1 & \frac{1}{2}(x_1^2 + x_2^2) - x_1 - 2x_2 \\
2x_1 + 3x_2 & \leq 6 \\
x_1 + 4x_2 & \leq 5 \\
-x_2 & \leq 0
\end{pmatrix}, \quad n = 2, m = 4
\]

Solution. Write standard form and input data
\[
\begin{align*}
2x_1 + 3x_2 & \leq 6 \\
x_1 + 4x_2 & \leq 5 \\
-x_2 & \leq 0
\end{align*}
\]

$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $c = \begin{pmatrix} -1 \\ -2 \end{pmatrix}$

$C^T = \begin{pmatrix} 2 & 1 \\ 3 & 4 \\ 0 & 1 \end{pmatrix}$

Use the formulas (29) and obtain
\[
\begin{pmatrix} 6 \\ 5 \\ 0 \end{pmatrix}, D = \begin{pmatrix} 13 & 14 & -2 & -3 \\ 14 & 17 & -1 & -4 \\ -2 & -1 & 1 & 0 \\ -3 & -4 & 0 & 1 \end{pmatrix}, d = \begin{pmatrix} -2 \\ -4 \\ 1 \\ 2 \end{pmatrix}
\]

The initial vector is $\theta$ and the results are .......
$k = 2 \quad 8/221 \quad 800/3757 \quad 0 \quad 0$
$k = 3 \quad 0 \quad 4/17 \quad 0 \quad 0$
$k = 4 \quad 0 \quad 4/17 \quad 0 \quad 0 \quad u^* \quad \text{STOP}$

$x^*=(13/17 \quad 18/17)^T$, $F(x^*)=-69/34$.

5. Nonlinear convex optimization and neural networks algorithms

5.1 Notations and mathematical background
Using the notations from [11], we denote
\[ \Omega = \{ x \in R^n / x \geq \theta \} \]
\[ \Omega_1 = \{ x \in R^n / p \leq x \leq r \} \]
\[ \Omega_2 = \{ x \in R^n / Cx = d, x \geq \theta \} \]
For any linear, quadratic or nonlinear optimization problem it is possible to define the Lagrange function. For models 5 and 6 we define
\[ L : R^{n+m} \rightarrow R, L(x,y) = F(x) + y^T (d - Cx) \]
where vector \( y \in R^m \) is the Lagrange multiplier and also the solution of associated dual problem. If \( x^* \) and \( y^* \) are the optimal solutions of both primal and dual problems, then from Khun–Tucker conditions or the complementary theorem the following inequalities hold [4], [11]
\[ L(x^*, y) \leq L(x, y^*) \leq L(x, y^*), \forall x \geq \theta, y \geq \theta' \]
In order to obtain the optimal solution of models 5, 6 we use the gradient vector of function \( L(x,y) \)
\[ \text{grad } L(x,y) = \nabla L(x,y) = \left( \frac{\partial L(x,y)}{\partial x}, \frac{\partial L(x,y)}{\partial y} \right) \]
\[ \frac{\partial L}{\partial x} (x,y) = \nabla F(x) - C^T y \in R^n \]
\[ \frac{\partial L}{\partial y} (x,y) = d - Cx \in R^m \]
\[ \nabla L(x,y) = \begin{pmatrix} \nabla F(x) - C^T y \\ d - Cx \end{pmatrix} \]

\( \nabla L(x,y) = \theta \in R^{n+m} \) \( \nabla L(x,y) = \theta \in R^{n+m} \) (45)
In any optimization problem we look for \( x \) so that
\[ x \geq \frac{\partial L}{\partial x} (x,y) \text{ and vector } x^* \text{ must have the property} \]
\[ x^* = \frac{\partial L}{\partial x} (x^*, y^*) \] (46)
Now appears an important question: how to implement the condition (46) and
\[ x \geq \frac{\partial L}{\partial x} (x,y), x \geq [\nabla F(x) - C^T y] \]
(47)
in a solving algorithm for our models? The idea from [11] is to construct the vector function (projection operator, projection function)
\[ [x - [\nabla F(x) - C^T y]]^+ = \begin{cases} x - [\nabla F(x) - C^T y], & x \geq [\nabla F(x) - C^T y] \\ 0, & x < [\nabla F(x) - C^T y] \end{cases} \]

Generally we denote this projection function by \([S]^+\), where
\[ S = (S_1 S_2 \cdots S_i \cdots S_n)^T, S \in R^n \]
\[ [S]^+ = ([S_1]^+ (S_2)^+ \cdots (S_n)^+]^T \]
\[ (S_i)^+ = \max \{0,S_i\}, i=1,n \] (48)

5.2 A neural network algorithm for solving a nonlinear convex optimization
As we have said, a neural network is an algorithm based on main elements used in neural network theory. In our case we use the following architecture: input level/input set, activation functions, output level/output set. The input level has \( n+m \) neurons. Idem for output level. The first \( n \) activation functions are sigmoid functions \( f_i \) and the last \( m \) are identity functions.

For model 5, the work [11] proposed the following dynamical equation of associated neural network
\[ \frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x - \alpha (\nabla F(x) - C^T y) \end{pmatrix}^+ - x \] (49)
where \( \lambda, \alpha \) are positive constants used as scaling factors (which accelerate the algorithm convergence). We denote \((x^* \ y^*)^T\) the numerical solution of the differential system (49). The vector \( x^* \) is the solution of optimization model 5 [11]. In the future we set \( \lambda = 1, \alpha = 1 \). The right hand side of (49) is called the derivatives matrix.

Remark 11. It is strange that [11] do not say anything how to implement the projection function \([\cdot]^+\) in solving the differential system (49) and nevertheless the journal IEEE Transactions on NN published such a work. Instead of this, [11] gives only a non-useful diagram as a numerical example.

In order to implement the projection function (49) we propose the using of the following sigmoid functions \( f_i \) (see (14)):
\[ S = x - \nabla F(x) + C^T y \] (50)
\[ f_i : R \rightarrow (b_i, a_i), f_i(S_i) = \frac{a_i e^{S_i/b_i} + b_i}{e^{S_i/b_i} + 1} \] (51)
where \( a_i, b_i \) are some real constants (real borders), given by the user. For \( x \geq \theta \) or \( \theta \leq x < \infty \) we take \( b_i = 0 \) and \( a_i \) a big real number.
The solving of differential system (49) is done by MathCAD subroutines \texttt{rkfixed} or \texttt{Rkadapt} (see section 3.6).

**Algorithm for model 5.**

Step 1. Input the initial vector \( u \). This is an arbitrary real vector so that
\[
\begin{align*}
  u &= \left( \begin{array}{c}
    x \\
    y
  \end{array} \right) = (x_1 \cdots x_n, y_1 \cdots y_m)^T \\
  u &= (u_0 u_1 \cdots u_n u_{n+1} \cdots u_{n+m-1}) \in R^{n+m}.
\end{align*}
\]

Step 2. Input the borders \( a_i, b_i \) and the slopes \( s_i, i = 1, n \).

Step 3. Use (50) and (51) and compute the vector \( S \) and the functions \( f_i(S_i), i = 1, n \) (or \( fSi \)).

Step 4. Construct the matrix of derivatives
\[
DModel5(t,u) = \begin{pmatrix}
  f_1(S_1) - u_0 \\
  \vdots \\
  f_n(S_n) - u_{n-1} \\
  d - C \cdots \\
  u_0 \\
  \vdots \\
  u_{n-1}
\end{pmatrix}
\] (52)

Step 5. We solve the differential system (53) by a MathCAD instruction, for example
\[
SolDModel5 = \text{rkfixed}(u,0,1,100,200,\text{DModel5})
\]

This instruction yields a table of numerical values.

**Remark 12.** The words \( DModel5 \) and \( SolDModel5 \) are at user’s disposal. Also, at user’s disposal are the following parameters:

\( u, a_i, b_i, s_i, t0, t1, N \) (see section 3.6).

**Remark 13.** The changing of parameters must produce a decreasing string of values of \( F(x) \).

Step 6. The solution \( u^* \) of differential system (52) was obtained when the numerical solution is stable by repetition. As a test of correctitude, the value \( F(x^*) \) have to be calculated.

**Algorithm for model 6**

In this model appear the linear bound constraints
\[
p \leq x \leq r, p = (p_1), r = (r_1), i = 1, n.
\]

In the above algorithm we change the step 2, where it is necessary to put \( b_i = p_i, \ a_i = r_i \) and, for example \( DModel6(t,u); SolDModel6 \).

Then the projection operator gives
\[
\begin{pmatrix}
  p_i, \ u_i < p_i \\
  f_i(u_i) = \begin{cases}
    u_i, & u_i \in [p_i, r_i] \\
    r_i, & u_i > r_i
  \end{cases}
\]

**Algorithm for model 7**

The differential system (49) and (50) become
\[
\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix}
  [x - \alpha(G(x) - C^T y)]^+ - x \\
  \alpha(d - Cx)
\end{pmatrix}
\] (53)

\[
S = x - G(x) + C^T y
\]

\( DMModel7(t,u); SolDMModel7 \).

All the steps from 1 to 6 hold.

**5.3 Nonlinear applications**

**Application 1.(5.3).** We have took this example from [11] and we changed a little bit the borders.

Find \( x \in R^n \) so that
\[
[\text{min}] F(x), Cx = d, x \in \Omega_1, \text{ where}
\]
\[
F(x) = \frac{3}{2} \left(x_1^2 + x_2^2\right) + 2 \left(x_3^2 + x_4^2\right) + 3x_1x_2 + \nonumber
\]
\[
+ 4x_3x_4 - 2x_1 - 3x_2 - \ln(x_1x_4)
\]
\[
C = \begin{pmatrix} 11000 \\ 0011 \end{pmatrix}, \ d = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \ p = (0.1 \ 0.0 \ 0.0 \ 0.1)^T
\]
\[
r = (5 \ 0.2 \ 0.2 \ 5)^T.
\]

The function \( F(x) \) could be written in matrix form, like that
\[
F(x) = x^T Qx + c^T x - \ln(x^T Q_1 x).
\]

**Solution.** We use the algorithm for model 6, with \( \lambda = 1, \alpha = 1 \). One obtains successively
\[
S = x - VF(x) + C^T y =
\]
\[
\begin{pmatrix}
  -2x_1 - 3x_2 + \frac{1}{x_1} + 2 + y_1 \\
  -3x_1 - 2x_2 + y_1 \\
  -3x_3 - 4x_4 + y_2 \\
  -4x_3 - 3x_4 + \frac{1}{x_4} + 3 + y_2
\end{pmatrix}
\]

\[
\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \frac{d}{dt} (u) = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{pmatrix} = \begin{pmatrix}
  f_1S1 - u_0 \\
  f_2S2 - u_1 \\
  f_3S3 - u_2 \\
  f_4S4 - u_3 \\
  1 - u_0 - u_1 \\
  1 - u_2 - u_3
\end{pmatrix}
\]

where \( f1S1 = \)
Proceedings of the 11th WSEAS International Conference on Sustainability in Science Engineering

\[ r_1 \exp \left[ s_1 \left( -2u_0 - 3u_1 + \frac{1}{u_0} + u_4 + 2 \right) \right] + p_1 \]
\[ = \frac{\exp \left[ s_1 \left( -2u_0 - 3u_1 + \frac{1}{u_0} + u_4 + 2 \right) \right] + 1}{\exp \left[ s_1 \left( -2u_0 - 3u_1 + \frac{1}{u_0} + u_4 + 2 \right) \right] + 1} \]
\[ f_{2S2} = \frac{r_2 \exp \left[ s_2 \left( -3u_0 - 2u_1 + u_4 \right) \right] + p_2}{\exp \left[ s_2 \left( -3u_0 - 2u_1 + u_4 \right) \right] + 1} \]
\[ f_{3S3} = \frac{r_3 \exp \left[ s_3 \left( -3u_2 - 4u_3 + u_5 \right) \right] + p_3}{\exp \left[ s_3 \left( -3u_2 - 4u_3 + u_5 \right) \right] + 1} \]
\[ f_{4S4} = \frac{r_4 \exp \left[ s_4 \left( -4u_2 - 3u_3 + \frac{1}{u_3} + u_5 + 3 \right) \right] + p_4}{\exp \left[ s_4 \left( -4u_2 - 3u_3 + \frac{1}{u_3} + u_5 + 3 \right) \right] + 1} \]

We used the following initial vector \( u \) and parameters:
\[
\begin{align*}
\mathbf{u} & = (0.11, 0.11, 0.11, 0.11, 0.11, 0.11)^T \\
s_1 & = 0.224, s_2 = 0.02, s_3 = 0.02, s_4 = 0.226.
\end{align*}
\]

\( DModel6(t, u) = \begin{pmatrix} f_{1S1} - u_0 \\ f_{2S2} - u_1 \\ f_{3S3} - u_2 \\ f_{4S4} - u_3 \\ 1 - u_0 - u_1 \\ 1 - u_2 - u_3 \end{pmatrix} \)

\( SolDModel6 = \text{rkfixed}(u, 0, 500, 500, DModel6) \)

By the above parameters we obtain the following table of numerical values:

<table>
<thead>
<tr>
<th>( u )</th>
<th>( u )</th>
<th>( u )</th>
<th>( u )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.001</td>
<td>0.1</td>
<td>0.1</td>
<td>1.001</td>
</tr>
<tr>
<td>1.001</td>
<td>0.1</td>
<td>0.1</td>
<td>1.001</td>
</tr>
<tr>
<td>1.001</td>
<td>0.1</td>
<td>0.1</td>
<td>1.001</td>
</tr>
</tbody>
</table>

... 

\( x^* = (1.001, 0.1, 0.1, 1.001)^T \)

\[ F(x^*) = 0.746. \]

**Remark 14.** ([11]) uses another border vectors \( p \) and \( r \) and one obtains the optimal solution \( x^* = (1, 0, 0, 1)^T \), without clear explications, but a lot of non-useful comments.

**Application 2.** ([5.3]). We have changed only the function \( F(x) \) in application 1.([5.3]).

Find \( x \in \mathbb{R}^n \) so that

\[ [\min] F(x), Cx = d, \ x \in \Omega_1, \] where

\[ F(x) = 5(x_1^2 + x_2^2) + 2(x_3^2 + x_4^2) + 10x_1x_2 + + 4x_3x_4 - 2x_1 - 3x_2 - \ln(10x_1x_4) \]
(or the corresponding matrix form).

**Solution.** The stable solution \( u^* \) is obtained by the following inputs

\[
\begin{align*}
\mathbf{s} & = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1)^T \\
\mathbf{s} & = (0.245, 0.245, 0.245, 0.245, 0.245, 0.245, 0.245, 0.245, 0.245, 0.245). \\
\mathbf{s} & = (0.245, 0.245, 0.245, 0.245, 0.245, 0.245, 0.245, 0.245, 0.245, 0.245).
\end{align*}
\]

\( SolDModel6 = \text{rkfixed}(u, 0, 500, 500, DModel6) \)

The stable solution \( u^* \) is

\[
\begin{align*}
\mathbf{u} & = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) \\
\mathbf{s} & = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) \\
\mathbf{s} & = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0).
\end{align*}
\]

\( x^* = (0.587, 0.102, 0.101, 0.908)^T \)

\[ F(x^*) = -1.162. \]

**Application 3.** ([5.3]). The variational inequality problem has

\[ \mathbf{G}(x) = \begin{pmatrix} 5x_1 + x_1^2 + x_2 + x_3 \\ 5x_1 + 3x_2^2 + 10x_2 + 3x_3 \\ 10x_1^2 + 8x_2^2 + 3x_2 + 3x_3^2 + x_2 \end{pmatrix} \]

\[ C = (1, 1)^T, \ d = 6. \]

**Solution.** \( S = x - \mathbf{G}(x) + C^T y \)

\[
\begin{align*}
& f_{1S1} = \frac{r_1 \exp \left[ s_1 \left( -4u_0 - u_0^2 - u_1 - u_2 + u_3 \right) \right] + p_1}{\exp \left[ s_1 \left( -4u_0 - u_0^2 - u_1 - u_2 + u_3 \right) \right] + 1} \\
& f_{2S2} = \frac{r_2 \exp \left[ s_2 \left( -5u_0 - 3u_1 - 9u_1 - 3u_2 + u_3 \right) \right] + p_2}{\exp \left[ s_2 \left( -5u_0 - 3u_1 - 9u_1 - 3u_2 + u_3 \right) \right] + 1} \\
& f_{3S3} = \frac{r_3 \exp \left[ s_3 \left( -10u_0^2 - 8u_1^2 - 3u_2 - 3u_2 + u_3 \right) \right] + p_3}{\exp \left[ s_3 \left( -10u_0^2 - 8u_1^2 - 3u_2 - 3u_2 + u_3 \right) \right] + 1}
\end{align*}
\]

The stable solution proceeds from parameters

\[
\begin{align*}
\mathbf{u} & = (0.0, 0.0, 0.0, 0.0) \\
\mathbf{s} & = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0).
\end{align*}
\]

The matrix of derivatives is...
\[
DModel_{63}(t,u) = \begin{pmatrix}
    f1S1 - u_0 \\
    f2S2 - u_1 \\
    f3S3 - u_2 \\
    6 - u_0 - u_1 - u_2
\end{pmatrix}
\]

\[
Sol(DModel_{63} = \text{rkfixed}(u,0,100,100))
\]

\[
\begin{align*}
    u_0 & = 4.5 \\
    u_1 & = 1.5 \\
    u_2 & = 0 \\
    u_3 & = 6
\end{align*}
\]

\[
x^* = (4.5, 1.5, 0.0)^T
\]

[Xia] has the same solution.

\[
Cx^* = 6
\]

6. Conclusions

The using of Neural Network in a quadratic optimization or nonlinear optimization appears as a surprise. The algorithms are very simple and the solving time is short. The algorithms based on the appropriate network is shown ([2], [11]) to be globally convergent. Nevertheless, finding the solution goes through the art of suitable choice of network parameters. This art is user’s duty and shows the user’s ability.

References


