Review
Transformation and Linearization Techniques in Optimization: A State-of-the-Art Survey

Mohammad Asghari 1, Amir M. Fathollahi-Fard 2, S. M. J. Mirzapour Al-e-hashem 3 and Maxim A. Dulebenets 4, *

1 Department of Industrial Engineering, Dalhousie University, 5269 Morris Street, Halifax, NS B3H 4R2, Canada; mohammad.asghari@dal.ca
2 Department of Electrical Engineering, École de Technologie Supérieure, University of Québec, Montréal, QC H3C 1K3, Canada; amirmohammad.fathollahifard.1@ens.etsmtl.ca
3 Department of Industrial Engineering and Management Systems, Amirkabir University of Technology (Tehran Polytechnic), Tehran 15875-4413, Iran; mirzapour@aut.ac.ir
4 Department of Civil & Environmental Engineering, College of Engineering, Florida A&M University-Florida State University (FAMU-FSU), 2525 Potsdam Street, Building A, Suite A124, Tallahassee, FL 32310-6046, USA
* Correspondence: mdulebenets@eng.famu.fsu.edu; Tel.: +1-(850)-410-6621

Abstract: To formulate a real-world optimization problem, it is sometimes necessary to adopt a set of non-linear terms in the mathematical formulation to capture specific operational characteristics of that decision problem. However, the use of non-linear terms generally increases computational complexity of the optimization model and the computational time required to solve it. This motivates the scientific community to develop efficient transformation and linearization approaches for the optimization models that have non-linear terms. Such transformations and linearizations are expected to decrease the computational complexity of the original non-linear optimization models and, ultimately, facilitate decision making. This study provides a detailed state-of-the-art review focusing on the existing transformation and linearization techniques that have been used for solving optimization models with non-linear terms within the objective functions and/or constraint sets. The existing transformation approaches are analyzed for a wide range of scenarios (multiplication of binary variables, multiplication of binary and continuous variables, multiplication of continuous variables, maximum/minimum operators, absolute value function, floor and ceiling functions, square root function, and multiple breakpoint function). Furthermore, a detailed review of piecewise approximating functions and log-linearization via Taylor series approximation is presented. Along with a review of the existing methods, this study proposes a new technique for linearizing the square root terms by means of transformation. The outcomes of this research are anticipated to reveal some important insights to researchers and practitioners, who are closely working with non-linear optimization models, and assist with effective decision making.

Keywords: linearization techniques; operations research analytics; transformation process; approximation; linear programming relaxation

1. Introduction

Many optimization problems in management science and operations research have been formulated in the non-linear programming form [1–5]. Due to their non-convex nature, there is no efficient method to locate the optimal solution for this kind of problem [6]. Finding a global optimum for a non-linear programming model in acceptable computational time is known as one of the major challenges in the optimization theory [7]. In comparison with non-linear programming problems, linear forms drive the solution process and have much lower computation time [8]. Therefore, linear programming (LP) forms of the optimization models are often recommended rather than solving integer
or non-linear forms [9,10]. The operations research techniques that have generally been adopted to solve optimization problems with non-linear terms can be classified into two broad groups, which include the following: (i) transformations in which the non-linear equations or functions are replaced by an exact equivalent LP formulation to create valid inequalities; and (ii) linear approximations which find the equivalent of a non-linear function with the least deviation around the point of interest or separate straight-line segments.

Transformation into the LP model generally requires particular manipulations and substitutions in the original non-linear model along with the implementation of valid inequalities. After solving the modified problem, the optimal values of the initial decision variables can be easily determined by reversing the transformation. Furthermore, approximation of complex non-linear functions with simpler ones is recognized as one of the common operations research techniques. In mathematics, a linear approximation of a function is an approximation (more precisely, an affine function) that relies on a set of linear segments for calculation purposes. Linear approximations are often adopted by finite difference methods, such as piecewise or first-order methods, to solve non-linear optimization models. A linear approximation to a known curve, as illustrated in Figure 1, can be obtained by dividing the curve and using linear interpolations between the points.

![Figure 1](image.jpg)

**Figure 1.** A non-linear function (blue) and its piecewise linear approximation (red).

Piecewise approximations play a major role in many areas of engineering and mathematics [11–13]. By adding extra variables and constraints, the piecewise linear approximation forms an alternative linear function that fits the original problem with a non-linear function. The specific aim of this technique is to estimate a one-variable single-valued function by a sequence of linear divisions. A piecewise linear approximation of the function \( f(x) \), which has been set on the interval \([a, b]\), approximates a close function \( g(x) \) that is represented by a set of linear segments over the same interval. \( g(x) \) can be represented as \( g(x) = c + dx \) for every \( x \) in \([a, b]\) [14]. The new linearity allows the previous non-linear optimization problem to be solved by common LP approaches, which are much easier to use and more efficient than their non-linear counterparts [15]. Some examples of piecewise linear approximations have been provided in Gajjar and Adil [16], Geißler et al. [17], Sridhar et al. [18], Andrade-Pineda et al. [19], and Stefanello et al. [20].

Taylor’s theorem approximates the output of a function \( f(x) \) around a given point, such as \( x = a \), by providing a \( k \)-times differentiable function and a polynomial of degree \( k \), which is known as the \( k \)-th order Taylor polynomial. In other words, the first-order Taylor polynomial provides a linear approximation of the real-valued function \( f(x) \) based on the value and slope of the function at \( x = b \), given that \( f(x) \) is differentiable on \([a, b]\) (or
[b, a]) and that a is close to b. This means that for a given twice continuously differentiable function of one real variable, the first-order Taylor polynomial can be represented as follows:

\[ f(x) = f(a) + f'(a)(x - a) + h(x)(x - a), \]

where \( h(x)(x - a) \) is the error term for the approximation. By removing the remainder term, the linear approximation of \( f(x) \) for \( x \) near the point \( a \) \((L_a(x))\) becomes \( y = f(a) + f'(a)(x - a) \) whose graph is a tangent line to the graph \( y = f(x) \) at the point \((a, f(a))\).

Figure 2 provides an illustration of the example graph of \( f(x) = e^x \) (blue) with its linear approximation \( L_a(x) = 1 + x \) (red) at \( a = 0 \). As \( x \) tends to be closer to \( a \), the error declines to zero much faster than \( f'(a)(x - a) \), which makes \( L_a(x) \approx f(x) \) a useful approximation.

The reformulation techniques, including transformation and linear approximation procedures, are able to generate a representation of the increasing degree of strength, albeit by increasing the size of the problems. Given the recent advances in LP, these techniques enhance the solvability of the problems within various exact or heuristic solution approaches by incorporating a tighter representation. A significant number of studies on the transformation process with LP representations have been conducted in the past, including Meyer [21], Jeroslow and Lowe [22,23], Balas [24], Johnson [25], Wolsey [26], Sherali and Adams [27,28], and Williams [29]. The performance of solution algorithms is directly associated with the tightness or strength of the adopted LP representations. Development of tight equivalents for separable or polynomial non-linear integer programming formulations can be achieved by generating valid inequalities [27,28,30–36].

The present study specifically concentrates on common operational research techniques, which have been widely used in the state-of-the-art approaches to convert non-linear components of optimization models into their LP equivalents. Although there are several sources that provide detailed information regarding various LP techniques [29,37], there is a lack of a holistic and comprehensive survey of the state-of-the-art transformation and linearization approaches for converting non-linear optimization models into their linearized forms. Along with the review of the existing methods, this study proposes a new technique for linearizing the square root terms by means of transformation to obtain tight LP relaxations. Furthermore, a new approach is presented to incorporate quadratic integers into LP mathematical formulations. The aforementioned contributions are expected to enhance the solvability of optimization models with non-linear terms and, ultimately, facilitate decision making.
The remaining sections of this manuscript are arranged in the following order. Section 2 focuses on the existing transformation approaches for different scenarios, including the multiplication of binary variables, multiplication of binary and continuous variables, multiplication of continuous variables, maximum/minimum operators, absolute value function, floor and ceiling functions, square root function, and multiple breakpoint function. Section 3 discusses linear approximations, including piecewise approximating functions and log-linearization via Taylor series approximation. Section 4 concludes this study, summarizes the key outcomes from the conducted work, and proposes some directions and opportunities for the future research.

2. Transformations

The techniques presented in this section are exact transformations of the original non-linear programs. In particular, the non-linear problems resulting from the multiplication of binary variables, multiplication of binary and continuous variables, multiplication of continuous variables, maximum/minimum operators, absolute value function, floor and ceiling functions, square root function, and multiple breakpoint function and their equivalent LP versions will be discussed in detail.

2.1. Multiplication of Binary Variables

In this section, a common linearization technique for the multiplication of binary variables is discussed. The suggested linearization method is based on some theoretical and numerical techniques [38]. Consider two binary variables \( x_i \) \((i \in \{1, \ldots, m\})\) and \( y_j \) \((j \in \{1, \ldots, n\})\). To linearize the term \( x_i \cdot y_j \), which results from multiplying the binary variables, we replace it with an additional binary variable:

\[
 z_{ij} := x_i \cdot y_j, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}. \tag{3}
\]

The model including the non-linear term can be linearized by adding some new constraints as follows:

\[
 z_{ij} \leq x_i, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}, \tag{4}
\]
\[
 z_{ij} \leq y_j, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}, \tag{5}
\]
\[
 z_{ij} \geq x_i + y_j - 1, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}, \tag{6}
\]
\[
 z_{ij} \in \{0, 1\}, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}. \tag{7}
\]

Table 1 examines the validity of these constraints and lists all possible scenarios by varying the values of binary variables \( x_i \) and \( y_j \).

When binary variables have power \( (x_i^p) \), without loss of generality, one can omit the power of \( p \) \((x_i := x_i^p)\), and it consequently can be linearized using the same technique. The extension to products of more than two variables is straightforward. In general, the multiplication of binary variables \( x_i^p \) \((k \in \{1, \ldots, K\}, i_k \in I_k = \{1, \ldots, m_k\})\) for \( K \geq 2 \) with different powers \( p \) can be linearized by replacing it with a new variable \( z_j \) \((z_j := \prod_{k=1}^{K} x_i^{p_k})\), where \( j \in \{i_1, \ldots, i_K\} \), and adding the following constraints:

\[
 z_j \leq x_{i_k}, \quad \forall k \in \{1, \ldots, K\}, \forall i \in I_k, \forall j \in \bigcup_{k=1}^{K} I_k, \tag{8}
\]
\[
 z_j \geq \sum_{k=1}^{K} x_{i_k} - (K - 1), \quad \forall k \in \{1, \ldots, K\}, \forall i \in I_k, \forall j \in \bigcup_{k=1}^{K} I_k, \tag{9}
\]
\[
 z_j \in \{0, 1\}, \quad \forall j \in \bigcup_{k=1}^{K} I_k. \tag{10}
\]
Table 1. All possible products of binary variables (\(z := x \cdot y\)).

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
<th>(x \cdot y)</th>
<th>Constraints</th>
<th>Imply</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(z \leq 0)</td>
<td>(z = 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(z \leq 0)</td>
<td>(z \leq -1)</td>
<td>(z \in {0, 1})</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>(z \leq 0)</td>
<td>(z = 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(z \leq 1)</td>
<td>(z \geq 0)</td>
<td>(z \in {0, 1})</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(z \leq 1)</td>
<td>(z = 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(z \leq 0)</td>
<td>(z \geq 0)</td>
<td>(z \in {0, 1})</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(z \leq 1)</td>
<td>(z = 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(z \leq 1)</td>
<td>(z \geq 1)</td>
<td>(z \in {0, 1})</td>
</tr>
</tbody>
</table>

2.2. Multiplication of Binary and Continuous Variables

In general, we can replace a multiplication of binary and continuous variables with a new variable, which is also subject to a number of new constraints. Let \(x_i\) be a binary variable for \(i \in \{1, \ldots, m\}\) and \(y_j\) be a continuous variable for which \(0 \leq y_j \leq u_j\) \((j \in \{1, \ldots, n\})\) holds. To linearize the bilinear term \(x_i \cdot y_j\), we replace it with the auxiliary variable \(z_{ij}\). Furthermore, the following constraint sets should also be imposed on the linear equivalent formulation, which force \(z_{ij}\) to take the value of \(x_i \cdot y_j\):

\[
\begin{align*}
    z_{ij} & \leq y_j, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}, \\
    z_{ij} & \leq u_j x_i, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}, \\
    z_{ij} & \geq y_j + u_j (x_i - 1), \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}, \\
    z_{ij} & \geq 0, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}.
\end{align*}
\]

where \(u_j\) can be replaced with a sufficiently large number. The validity of these constraints can be checked by examining Table 2, in which all possible scenarios are listed. The studies conducted by Asghari and Mirzapour Al-e-hashem [39], Asghari et al. [40], and Mojtahedi et al. [41] can serve as relevant examples of applying this method.

Table 2. All possible products of binary and continuous variables (\(z := x \cdot y\)).

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
<th>(x \cdot y)</th>
<th>Constraints</th>
<th>Imply</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(m: 0 \leq m \leq u)</td>
<td>0</td>
<td>(z \leq m)</td>
<td>(z = 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(z \leq 0)</td>
<td>(z \geq m - u)</td>
<td>(z \geq 0)</td>
</tr>
<tr>
<td>1</td>
<td>(m: 0 \leq m \leq u)</td>
<td>(m)</td>
<td>(z \leq m)</td>
<td>(z = m)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(z \leq u)</td>
<td>(z \geq m)</td>
<td>(z \geq 0)</td>
</tr>
</tbody>
</table>

2.3. Multiplication of Two Continuous Variables

This subsection discusses an effective method for linearizing equations that incorporate a product of continuous variables. Linearization of multiplication of continuous variables can be complex. Solving such a function can become extremely difficult. The AIMMS
Modelling Guide [37] provides a hint for bounded variables by which the product of two continuous variables can be transformed into a separable form. We assume that term \( x_1 \cdot x_2 \) must be converted. First of all, we define two new continuous variables \( y_1 \) and \( y_2 \) as follows:

\[
y_1 = \frac{1}{2} (x_1 + x_2),
\]

\[
y_2 = \frac{1}{2} (x_1 - x_2).
\]

The product \( x_1 \cdot x_2 \) can be now replaced with the below separable function:

\[
y_1^2 - y_2^2 := x_1 \cdot x_2.
\]

which can be linearized by using the technique of the piecewise approximation as stated in Section 3.1 [29]. Note that one can eliminate the non-linear function at the cost of having to approximate the objective. If \( l_1 \leq x_1 \leq u_1 \) and \( l_2 \leq x_2 \leq u_2 \), then the lower and upper bounds on \( y_1 \) and \( y_2 \) are:

\[
\frac{1}{2} (l_1 + l_2) \leq y_1 \leq \frac{1}{2} (u_1 + u_2),
\]

\[
\frac{1}{2} (l_1 - u_2) \leq y_2 \leq \frac{1}{2} (u_1 - l_2).
\]

It should be noted that the product \( x_1 \cdot x_2 \) can be substituted with a single variable \( z \) whenever (i) one of the variables is not referenced in any other term except in the products of the above form, and (ii) the lower bounds \( l_1 \) and \( l_2 \) are non-negative. Suppose \( x_1 \) is such a variable (not used in any other terms). Then, the non-linear term \( x_1 \cdot x_2 \) can be replaced with \( z \) just by adding the following constraint:

\[
l_1 \cdot x_2 \leq z \leq u_1 \cdot x_2.
\]

Once the resulting mathematical formulation is solved in terms of \( z \) and \( x_2 \), it is required to calculate \( x_1 = \frac{z}{x_2} \) whenever \( x_2 > 0 \). \( x_1 \) is undetermined when \( x_2 = 0 \). The extra constraints on \( z \) ensure that \( l_1 \leq x_1 \leq u_1 \) when \( x_2 > 0 \).

2.4. Maximum/Minimum Operators

The maximum and minimum operators are viewed as explicit non-linear terms. These terms can also be linearized to efficiently solve the optimization model in which they are directly used [42]. Assume there is a general non-linear structure in the form of \( \max\{x_i\} \), where \( i = \{1, \ldots, n\} \). This structure can be further converted to an equivalent mathematical model after adding a new continuous variable \( z \), a set of new binary variables \( y_i \), and introducing supplementary constraint sets (22) to (25).

\[
z := \max_{i \in I} \{x_i\},
\]

\[
z \geq x_i, \quad \forall i \in \{1, \ldots, n\},
\]

\[
z \leq x_i + m \cdot y_i, \quad \forall i \in \{1, \ldots, n\},
\]

\[
\sum_{i=1}^{n} y_i \leq n - 1,
\]

\[
y_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}.
\]

where \( m \) is a sufficiently large number. Constraints (22) assure that \( z \) is greater than all \( x_i \). Constraints (23) and (24) are applied to prevent \( z \) from becoming infinite. In particular, constraints (23) and (24) are used to assure that for only one \( i \), \( z \) has to be less than or equal to \( x_i \). Constraints (23) and (24) do not have to be applied when the objective function is of a minimization type. However, constraints (23) and (24) are required when the objective function is of a maximization type and/or the considered optimization model has multiple
objective functions. Constraints (25) denote the integrality restrictions on the values of binary variables \(y_i\). For the non-linear term \(\min_{i \in I} \{x_i\}\), where \(I = \{1, \ldots, n\}\), Equations (22) and (23) have to be altered as follows:

\[
\begin{align*}
    z &\leq x_i, \quad \forall i \in \{1, \ldots, n\}, \\
    z &\geq x_i - m \cdot y_i, \quad \forall i \in \{1, \ldots, n\}.
\end{align*}
\]  

(26)  

(27)

where \(z := \min_{i \in I} \{x_i\}\) and constraints (24) and (25) stay unaltered. In this case, constraints (24) and (27) do not have to be applied when the objective function is maximization.

2.5. Absolute Value Function

One of the special cases in non-linear programming is optimization problems using the absolute value function, on which it is extremely hard to apply standard optimization methods. Operating on the absolute value expression is relatively difficult because it is sometimes not a continuously distinguishable function. However, it is possible to avoid these difficulties and solve the problem using LP procedures by simply manipulating the absolute values \([43–47]\). If a linear function locates in an absolute value function, then we can alternatively use a valid inequality instead of the absolute value. This means that \(z = |f(x)|\) can be effectively converted to two linear expressions if the function \(f(x)\) is linear itself. For the simplest example, \(z = |x|\), the function can practically be reformulated by combining two piecewise functions: \(z = x\) if \(x \geq 0\) and \(-z = x\) if \(x < 0\). The aforementioned procedure is the basis of running LP formulations that contain the absolute value functions.

2.5.1. Absolute Value in Constraints

In the case \(|f(x)| \leq z\), we are able to reformulate the expression as the combination of \(f(x) \leq z\) and \(-f(x) \leq z\). This relation is demonstrated in Figure 3 using a number line. The figure depicts that the two formulations, the above two linear functions, as well as the absolute value function, are equivalent. The same logic can be used to reformulate \(|f(x)| \geq z\) as \(f(x) \geq z\) and \(-f(x) \geq z\), or \(|f(x)| + g(y) \leq z\) into \(f(x) + g(y) \leq z\) and \(-f(x) + g(y) \leq z\) \([48]\).

![Figure 3. Number line representing the expression \(|f(x)| \leq z\).](image)

2.5.2. Absolute Value in the Objective Function

In certain cases, we can also reformulate the absolute value used in the objective function to become linear. A model can be easily transformed to be solved using LP when its objective function is a maximization in the form of \(-|f(x)| + g(y)\) or a minimization in the form of \(|f(x)| + g(y)\). Unfortunately, when the objective function of the model is a maximization in the form of \(|f(x)| + g(y)\) or a minimization in the form of \(-|f(x)| + g(y)\), the model cannot be turned into a standard linear form, and instead must be solved using mixed-integer LP. In the first two cases, as mentioned in Mangasarian \([44]\) and Caccetta et al. \([47]\), the model can be reformulated by substituting a new variable \(z\) with \(|f(x)|\) within the original objective function, and adding two extra constraints \(f(x) \leq z\) and \(-f(x) \leq z\).

2.5.3. Minimizing the Sum of Absolute Deviations

The transformation approach for minimizing the sum of absolute deviations presented herein was introduced by Ferguson and Sargent \([49]\). Let the deviations be represented by \(x_i = b_i - \sum_{j} a_{ij} y_j\), where \(i\) is the \(i\)th observation \((i \in \{1, \ldots, m\})\), \(b_i\) is an observation, and \(x_i\) gives the deviation. The objective of the mathematical formulation aims to minimize the deviation and can be formulated in the following basic form:
and the linear constraints are
\[ x_i + \sum_{j=1}^{n} a_{ij} y_j = b_i, \quad \forall i \in \{1, \ldots, m\}, \]
\[ x_i, y_j \in \mathbb{R}, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}. \]

The absolute value function generates the non-linearity in this form. However, the model can be rewritten as an LP model by replacing \( x_i \) with \( x_i^+ - x_i^- \) (where \( x_i^+ \) and \( x_i^- \) are positive variables), and the model can be reformulated as:
\[
\min \sum_{i=1}^{m} |x_i|, \quad \text{(28)}
\]
\[
\text{s.t.}
\]
\[
x_i^+ - x_i^- + \sum_{j=1}^{n} a_{ij} y_j = b_i, \quad \forall i \in \{1, \ldots, m\}, \quad \text{(29)}
\]
\[
x_i = x_i^+ - x_i^-, \quad \forall i \in \{1, \ldots, m\}, \quad \text{(30)}
\]
\[
x_i^+, x_i^- \geq 0, \quad \forall i \in \{1, \ldots, m\}, \quad \text{(31)}
\]
\[
x_i, y_j \in \mathbb{R}, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}. \quad \text{(32)}
\]

At the optimal solution, it can be proven that \( x_i^+ x_i^- = 0 \). Therefore, the model is reformulated to a linear programming form, as \( |x_i| = x_i^+ + x_i^- \geq 0 \). The final form of the problem is
\[
\min \sum_{i=1}^{m} (x_i^+ + x_i^-), \quad \text{(33)}
\]
\[
\text{s.t.}
\]
\[
x_i^+ - x_i^- + \sum_{j=1}^{n} a_{ij} y_j = b_i, \quad \forall i \in \{1, \ldots, m\}, \quad \text{(34)}
\]
\[
x_i = x_i^+ - x_i^-, \quad \forall i \in \{1, \ldots, m\}, \quad \text{(35)}
\]
\[
x_i^+, x_i^- \geq 0, \quad \forall i \in \{1, \ldots, m\}, \quad \text{(36)}
\]
\[
x_i, y_j \in \mathbb{R}, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}. \quad \text{(37)}
\]

2.5.4. Minimizing the Maximum of Absolute Values

In some cases, such as the evaluation of the maximum forecast error by applying the Chebyschev criterion, the problem aims to minimize the largest absolute deviation rather than the sum. Such a formulation can be expressed using Equations (41) to (43).
\[
\min \max_i |x_i|, \quad \text{(38)}
\]
\[
\text{s.t.}
\]
\[
x_i + \sum_{j=1}^{n} a_{ij} y_j = b_i, \quad \forall i \in \{1, \ldots, m\}, \quad \text{(39)}
\]
\[
x_i, y_j \in \mathbb{R}, \quad \forall i \in \{1, \ldots, m\}, \forall j \in \{1, \ldots, n\}. \quad \text{(40)}
\]

where variable \( x_i \) indicates the deviation for the \( i \)th observation and \( y_j \) indicates the \( j \)th variable in the forecasting equation. The constraints have been described in the previous subsection. To solve this model, variable \( x \) is introduced, which satisfies the following two inequalities:
\[
x \geq b_i - \sum_{j=1}^{n} a_{ij} y_j, \quad \forall i \in \{1, \ldots, m\}, \quad \text{(41)}
\]
\[
x \geq -(b_i - \sum_{j=1}^{n} a_{ij} y_j), \quad \forall i \in \{1, \ldots, m\}. \quad \text{(42)}
\]
The inequalities (44) and (45) guarantee that \( x \) is greater than or equal to the largest \( |x_i| \). Therefore, as stated by McCarl and Spreen [50], the original formulation can be represented in the following form:

\[
\begin{align*}
\text{Min } & \quad x, \\
\text{s.t. } & \quad -x - \sum_{j=1}^{n} a_{ij} y_j \leq -b_i, \quad \forall i \in \{1, \ldots, m\}, \\
& \quad -x + \sum_{j=1}^{n} a_{ij} y_j \leq b_i, \quad \forall i \in \{1, \ldots, m\}, \\
& \quad x \geq 0, \\
& \quad y_j \in \mathbb{R}, \quad \forall j \in \{1, \ldots, n\}.
\end{align*}
\]

2.6. Floor and Ceiling Functions

The floor function is a mathematical function that takes a certain real number \( x \) as an input and returns the greatest integer that is less than or equal to \( x \) as an output. In a similar fashion, the ceiling function is a mathematical function that rounds \( x \) to the least integer that is greater than or equal to \( x \). Let \( \lfloor \rfloor \) denote the floor integer function and consider the non-linear equation \( \lfloor f(x) \rfloor \). The value of function \( f(x) \) can be represented as \( f(x) = y + r \), where \( y \) is the integral part of \( f(x) \) and \( 0 \leq r < 1 \). Therefore, the floor function \( \lfloor f(x) \rfloor \) can be linearized by replacing it with the integer variable \( y \) (\( y \lfloor f(x) \rfloor \)) and adding the following constraints:

\[
\begin{align*}
y \leq f(x) < y + 1, \\
y \in \mathbb{Z}.
\end{align*}
\]

Equation (52) is the integrality constraint. A similar approach can be used to linearize the ceiling function \( \lceil f(x) \rceil \) by replacing it with the integer variable \( y \) (\( y \lceil f(x) \rceil \)) and adding the following constraints to the problem:

\[
\begin{align*}
y - 1 \leq f(x) < y, \\
y \in \mathbb{Z}.
\end{align*}
\]

2.7. Square Root Function

A square root of a certain number \( x \) is defined in mathematics using another number \( y \) such that \( y^2 = x \); alternatively, a number \( y \) that has a squared value (i.e., the resulting value of multiplying a given number by itself or \( y \cdot y \)) of \( x \) [51]. Each real-valued non-negative number \( x \) has a unique non-negative square root (that is also referred to as the principal square root), which can be represented by mathematical notation \( \sqrt{x} \). The linearization of a square root function, to the authors’ knowledge, has not yet been studied in the literature (or been applied in practice). Several studies addressing such a function only used the Taylor-series approximation method by writing the equation of the line tangent to the function at a given constant. Kwon and Draper [52] and Del Moral and Nicolas [53] are examples of Taylor-series expansion algorithms developed for the square root function.

This section of the manuscript proposes a new technique for linearizing the square root terms. The suggested linearization method is based on some numerical and theoretical techniques in the area of mixed-integer programming that were presented by Rahil [38]. Consider the square root \( \sqrt{f(x)} \) which is an explicit non-linear term named here as radical. To linearize the radical term, it can be replaced with a new positive integer variable \( A \) as below:

\[
\begin{align*}
A &= \lfloor \text{radical} \rfloor, \\
A &\in \mathbb{Z}^+.
\end{align*}
\]
where $\lceil \cdot \rceil$ is the ceiling bracket sign that rounds up radical to the nearest integer that is greater than or equal to the expression. In Equation (55), radical is a positive real number, and $A$ is a positive integer number. Thus, $A$ ignores the fraction part of radical (if there is any fraction) and approximates it with a strictly lower than one unit error. This amount of error can be negligible, especially when radical is a very large number. By using Theorem 1, the value of $A$ can be converted into its binary equivalent.

**Theorem 1.** Assume that $w$ is an integer variable. Then,

$$ w = \sum_{i=0}^{n-1} 2^i y_i + (u - 2^n + 1) y_n, $$

(57)

where $y_i$ are binary variables, $u$ denotes the upper bound of $w$, and $n$ is set to $\lfloor \log_2 (u + 1) \rfloor$.

For example, if $u = 45$, any positive integer number $w$ less than 45 can be written as binary using the following equation:

$$ w = 2^0 y_0 + 2^1 y_1 + 2^2 y_2 + 2^3 y_3 + 2^4 y_4 + (45 - 2^5 + 1) y_5 $$

$$ = 1 + 2 y_1 + 2^2 y_2 + 2^3 y_3 + 2^4 y_4 + 14 y_5. $$

(58)

It should also be noted that integer numbers greater than 45 cannot be constructed by Equation (58). Therefore, the upper bound condition, $u$, is never violated. Proof of Theorem 1 is provided in Appendix A. The upper bound can be determined by calculating the maximum function $f(x) = \max_x \{ f(x) \}$ as follows:

$$ u = \sqrt{f(x)}. $$

(59)

As shown in Equation (53), $A = \lceil \sqrt{f(x)} \rceil$, then $\sqrt{f(x)} \leq A$. To calculate the upper bound of function $f(x)$, we rise the two sides of this inequality to the power of 2:

$$ f(x) \leq A^2. $$

(60)

At this point, we can use Theorem 2 to precisely simulate the quadratic term $A^2$.

**Theorem 2.** Assume $w$ is an integer variable that has an upper bound of $u$. After that, $w^2$ can be further linearized based on the following equation:

$$ w^2 = \sum_{i=0}^{n-1} 2^{2i} y_i + \sum_{j>i}^{n-1} 2^{i+j+1} z_{ij} + \beta \cdot \sum_{i=0}^{n-1} 2^{i+1} z_{in} + \beta^2 y_n, $$

(61)

where $y_i$ and $z_{ij}$ are binary variables, $\beta = (u - 2^n + 1)$, and $n$ is set to $\lfloor \log_2 (u + 1) \rfloor$.

The proof of Theorem 2 is provided in Appendix B. The linearization of the quadratic integers is presented in Appendix C. By utilizing Theorems 1 and 2, we obtain the linearized equivalent of a square root by substituting $\sqrt{f(x)}$ by $A$ and adding the following constraints:

$$ f(x) \leq \sum_{i=0}^{n-1} 2^{2i} y_i + \sum_{j>i}^{n-1} 2^{i+j+1} z_{ij} + (u - 2^n + 1) \cdot \sum_{i=0}^{n-1} 2^{i+1} z_{in} + (u - 2^n + 1)^2 y_n, $$

(62)

$$ A = \sum_{i=0}^{n-1} 2^{2i} y_i + (u - 2^n + 1) y_n, $$

(63)

$$ z_{ij} \leq y_{ij}, \quad \forall i, j \in \{0, \ldots, n\}, $$

(64)
\[ z_{ij} \leq y_j, \quad \forall i, j \in \{0, \ldots, n\}, \quad (65) \]
\[ z_{ij} \geq y_i + y_j - 1, \quad \forall i, j \in \{0, \ldots, n\}, \quad (66) \]
\[ y_i, z_{ij} \in \{0, 1\}, \quad \forall i, j \in \{0, \ldots, n\}, \quad (67) \]
\[ A \in \mathbb{Z}^+. \quad (68) \]

To show how Equations (61)–(68) can be applied in a specific case, let us estimate the square root \( \sqrt{x} \) with an upper bound of 103. \( n \) is calculated accordingly as \( \lfloor \log_2(103 + 1) \rfloor = 6 \). The square root function can be linearized by replacing it with the integer variable \( A \) and adding the following constraints to the problem:

\[
x \leq \sum_{i=0}^{5} 2^{2i} y_i + \sum_{i=0}^{4} \sum_{j>i} 2^{i+j+1} z_{ij} + 40 \sum_{i=0}^{5} 2^{i+1} z_{i,6} + 1600 y_6, \quad (69)\]
\[
A = \sum_{i=0}^{5} 2^i y_i + 40 y_6, \quad (70)\]
\[
z_{ij} \leq y_i, \quad \forall i, j \in \{0, \ldots, 6\}, \quad (71)\]
\[
z_{ij} \leq y_j, \quad \forall i, j \in \{0, \ldots, 6\}, \quad (72)\]
\[
z_{ij} \geq y_i + y_j - 1, \quad \forall i, j \in \{0, \ldots, 6\}, \quad (73)\]
\[
y_i, z_{ij} \in \{0, 1\}, \quad \forall i, j \in \{0, \ldots, 6\}, \quad (74)\]
\[ A \in \mathbb{Z}^+. \quad (75)\]

### 2.8. Multiple Breakpoint Function

In this section, we describe two linearization techniques for multiple breakpoint functions based on the information presented in Tsai [54] and Mirzapour Al-e-hashem et al. [55]. Suppose there is a general continuous multiple breakpoint function that can be defined as follows:

\[
f(x) = \begin{cases} 
  a_1 \cdot x + b_1, & \text{if } c_0 \leq x \leq c_1 \\
  a_2 \cdot x + b_2, & \text{if } c_1 \leq x \leq c_2 \\
  \vdots & \vdots \\
  a_n \cdot x + b_n, & \text{if } c_{n-1} \leq x \leq c_n 
\end{cases} \quad (76)
\]
\[ x \in \mathbb{R}. \quad (77)\]

Based on the methodology suggested by Tsai [54], the equivalent valid inequality of Equation (76) can be further simplified to the following form:

\[
f(x) = \sum_{i=1}^{n} t_i \cdot (a_i \cdot x + b_i), \quad (78)\]
\[ s.t. \]
\[
\sum_{i=1}^{n} c_{i-1} \cdot t_i \leq x \leq \sum_{i=1}^{n} c_i \cdot t_i, \quad (79)\]
\[
\sum_{i=1}^{n} t_i = 1, \quad (80)\]
\[ t_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}. \quad (81)\]

As can be seen, this formulation contains an explicit non-linear term \( t_i \cdot x \). As proven by Tsai [54], an equivalent linear equation for \( z = t \cdot g(x); t \in \{0, 1\} \) can be reformulated as:
\[ g(x) - (1 - t) \cdot m \leq z \leq g(x) + (1 - t) \cdot m, \] (82)

\[ -t \cdot m \leq z \leq t \cdot m, \] (83)

\[ t \in \{0, 1\}. \] (84)

where \( m \) is a sufficiently large number.

Let us consider \( a_i x + b_i \) as \( g(x) \); then, Tsai’s techniques can be rewritten as follows:

\[ f(x) = \sum_{i=1}^{n} z_i, \] (85)

s.t.

\[ \sum_{i=1}^{n} c_{i-1} t_i \leq x \leq \sum_{i=1}^{n} c_i t_i, \] (86)

\[ \sum_{i=1}^{n} t_i = 1, \] (87)

\[ a_i x + b_i - (1 - t_i) \cdot m \leq z_i, \quad \forall i \in \{1, \ldots, n\}, \] (88)

\[ a_i x + b_i + (1 - t_i) \cdot m \geq z_i, \quad \forall i \in \{1, \ldots, n\}, \] (89)

\[ -t_i \cdot m \leq z_i \leq t_i \cdot m, \quad \forall i \in \{1, \ldots, n\}, \] (90)

\[ t_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \] (91)

\[ z_i \in \mathbb{R}, \quad \forall i \in \{1, \ldots, n\}. \] (92)

Mirzapour Al-e-hashem et al. [55] proposed another linearization technique for a multiple breakpoint function. The authors have shown that the multiple breakpoint function \( f(x) \) can be linearized by introducing some binary variables \( t_i \) and also converting variable \( x \) to \( n \) independent variables \( x_i \), where \( x = \sum x_i \). Thus, Equation (76) can be rewritten as follows:

\[ f(x) = \begin{cases} a_1 x_1 + b_1, & \text{if } c_0 \leq x_1 \leq c_1 \\ a_2 x_2 + b_2, & \text{if } c_1 \leq x_2 \leq c_2 \\ \vdots & \vdots \\ a_n x_n + b_n, & \text{if } c_{n-1} \leq x_n \leq c_n \end{cases}. \] (93)

As proven by Mirzapour Al-e-hashem et al. [55], the linear equivalent mathematical structure of \( f(x) \) can be written by introducing new constraints as follows:

\[ f(x) = \sum_{i=1}^{n} (a_i x_i + b_i \cdot t_i), \] (94)

s.t.

\[ c_{i-1} t_i \leq x_i \leq c_i t_i, \quad \forall i \in \{1, \ldots, n\}, \] (95)

\[ \sum_{i=1}^{n} x_i = x, \] (96)

\[ \sum_{i=1}^{n} t_i = 1, \] (97)

\[ t_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \] (98)

\[ x_i \in \mathbb{R}, \quad \forall i \in \{1, \ldots, n\}. \] (99)

For a non-continuous multiple breakpoint function of type:
f(x) = \begin{cases} a_1 \cdot x + b_1, & \text{if } x \leq c_1 \\ a_2 \cdot x + b_2, & \text{if } c_1 < x \leq c_2 \\ \vdots & \vdots \\ a_n \cdot x + b_n, & \text{if } c_{n-1} < x \\ \end{cases}, \quad (100)

x \in \mathbb{R}. \quad (101)

The Mirzapour’s technique can be used by modifying Equation (95) as follows:

\begin{align*}
\left( c_{i-1} + \frac{1}{m} \right) t_i & \leq x_i \leq c_i \cdot t_i, & \forall i \in \{2, \ldots, n-1\}, \\
x_1 & \leq c_1 \cdot t_1, \\
\left( c_{n-1} + \frac{1}{m} \right) t_n & \leq x_n,
\end{align*}

where \( m \) is a sufficiently large number.

Comparing the two aforementioned methodologies, the same amount of binary and continuous variables is used. However, the method proposed by Mirzapour Al-e-hashem et al. [55] requires five new different classes of constraints for the initial model. On the contrary, the Tsai’s technique imposes seven new different classes of constraints for the original model.

3. Approximate Linearization Methods

Approaches to non-linear programming typically use approximation techniques that may be either iterative or non-iterative (i.e., the ones that require just one iteration). This section discusses the main approximation techniques that can be implemented for the majority of non-linear problems. The considered approximation techniques could be divided into two broad groups, including (i) piecewise linear approximation techniques; and (ii) log-linearization via Taylor series approximation. For more information regarding alternative approximation techniques, the interested readers can refer to Dembo [56], McCarl and Tice [57], and McCarl and Onal [58].

3.1. Piecewise Linear Approximation

In many studies over the past decades, piecewise linear approximation (PLA) techniques have been used to convert the non-linear LP models into their linear forms or mixed-integer convex optimization problems to obtain approximate global optimal solutions. To reformulate the original non-linear optimization problem, new variables of binary and continuous nature along with extra constraints are generally applied in the transformation process. The additional variables and constraint sets typically improve the effectiveness of obtaining solutions for the converted problem. The next sections of the manuscript present an overview of common PLAs and analyze their computational efficiency.

3.1.1. Formulations

Consider a general non-linear function \( f(x) \) of a single variable \( x \), which is within the interval \([a_0, a_n]\). Such a continuous function has the advantage that we can approximate its non-linear expressions to piecewise linear ones as commonly used in the non-linear programming literature [59–62]. For comparing PLA techniques, interested readers can refer to the computational results reported by Li et al. [59] and Lin et al. [63].

- **Method 1**

  First, divide \( f(x) \) into \( n \) separate segments and define \( a_i \) (\( i \in \{0, 1, \ldots, n\} \)) as the breakpoints of \( f(x) \), \( a_0 < a_1 < \ldots < a_n \). Then, we can approximately linearize the non-linear function \( f(x) \) over the interval \([a_0, a_n]\) as follows:
$L(f(x)) = \sum_{i=0}^{n} f(a_i) \cdot y_i,$  

s. t.

\[ \sum_{i=0}^{n} a_i \cdot y_i = x, \quad (105) \]

\[ \sum_{i=0}^{n} y_i = 1, \quad (106) \]

\[ y_i \leq t_{i-1} + t_i, \quad \forall i \in \{1, \ldots, n-1\}, \quad (107) \]

\[ y_n \leq t_{n-1}, \quad (108) \]

\[ y_i \in \mathbb{R}^+, \quad \forall i \in \{0, \ldots, n\}. \quad (109) \]

where only two adjacent $y_i$ are allowed to be non-zero.

Figure 4 illustrates the piecewise linearization of $f(x)$. The above terms include $n$ extra binary variables $t_0$, $t_1$, $\ldots$, $t_{n-1}$ and $n+1$ new continuous variables $y_0$, $y_1$, $\ldots$, $y_n$. The number of these newly added variables increases with the number of breakpoints ($n+1$), which leads to an exponential increase in computational time. The more linear segments there are, the more accurate the approximation will be at the expense of increasing the computational complexity.

**Method 2**

Li and Yu [64] introduced a method for the global optimization of non-linear mathematical models in which the objective along with the constraint sets may be non-convex. First, a univariate mathematical function is formulated via a piecewise linear function using a sum of absolute expressions. Denote $s_i (i \in \{0, 1, \ldots, n-1\})$ as the slopes of each line between $a_i$ and $a_{i+1}$, computed using Equation (114):

\[ s_i = \frac{f(a_{i+1}) - f(a_i)}{a_{i+1} - a_i}, \quad \forall i \in \{0, \ldots, n-1\}. \quad (114) \]
An equivalent piecewise linear form of non-linear function \( f(x) \) can then be reformulated as follows:

\[
L(f(x)) = f(a_0) + s_0 \cdot (x - a_0) + \sum_{i=1}^{n-1} \frac{s_i - s_{i-1}}{2} \cdot (|x - a_i| + x - a_i).
\]  

(115)

If the successive slopes of the PLA are non-decreasing \((s_i - s_{i-1} \geq 0)\) in the interval \([a_{i-1}, a_i]\), \( f(x) \) is convex. Otherwise, \( f(x) \) is a concave (non-convex) function when these slopes are increasing. After linearization of the absolute term, Li and Yu [64] included additional binary variables \( t_i \) to convert the non-convex model to a piecewise linear form as follows:

\[
L(f(x)) = f(a_0) + s_0 \cdot (x - a_0) + \sum_{i=1}^{n-2} (s_i - s_{i-1}) \cdot \left(x - a_i + \sum_{j=0}^{i-1} d_j\right)
\]  

\[
+ \frac{1}{2} \sum_{i=1}^{n-1} (s_i - s_{i-1}) \cdot (x - a_i + 2a_i t_i - 2 z_i),
\]  

s.t.

\[x + \sum_{i=0}^{n-2} d_i \geq a_{n-1},\]

(117)

\[d_i \leq a_{i+1} - a_i, \quad \forall i \in \{0, \ldots, n-1\} \text{ and } s_i > s_{i-1},\]

(118)

\[x + u(t_i - 1) \leq z_i, \quad \forall i \in \{0, \ldots, n-1\} \text{ and } s_i < s_{i-1},\]

(119)

\[t_i \in \{0,1\}, \quad \forall i \in \{0,\ldots, n-1\},\]

(120)

\[d_i, z_i \in \mathbb{R}^+, \quad \forall i \in \{0,\ldots, n-1\}.\]

(121)

where \( u \) is the upper bound of \( x \).

Compared to Method 1, which used binary variables for all parts, the binary variables applied by the second approach have only been used to linearize the non-convex intervals of \( f(x) \). Thus, the second method generally employs fewer binary variables than Method 1.

- **Method 3**

Another representing form of the piecewise approximating function has been used in the studies conducted by Li and Tsai [65], Topaloglu and Powell [66], Padberg [67], Li [68], and Croxton et al. [69]. The equivalent function can be formulated as demonstrated below:

\[x \geq a_i - (a_n - a_0)(1 - t_i), \quad \forall i \in \{0, \ldots, n-1\},\]

(122)

\[x \leq a_{i+1} + (a_n - a_0)(1 - t_i), \quad \forall i \in \{0, \ldots, n-1\},\]

(123)

\[f(x) \geq f(a_i) + s_i \cdot (x - a_i) - m(1 - t_i), \quad \forall i \in \{0, \ldots, n-1\},\]

(124)

\[f(x) \leq f(a_i) + s_i \cdot (x - a_i) + m(1 - t_i), \quad \forall i \in \{0, \ldots, n-1\},\]

(125)

\[s_i = \frac{f(a_{i+1}) - f(a_i)}{a_{i+1} - a_i}, \quad \forall i \in \{0, \ldots, n-1\},\]

(126)

\[
\sum_{i=0}^{n-1} t_i = 1,
\]

(127)

\[t_i \in \{0,1\}, \quad \forall i \in \{0, \ldots, n-1\}.\]

(128)

where \( m \) is a sufficiently large constant.

When \( n + 1 \) breakpoints are used, the above approach requires extra \( n \) binary variables and \( 4n \) new constraint sets to model a piecewise linear function.
Method 4

More linear pieces in the piecewise linear programming enhance the accuracy of the approximating function, but in the meantime, the execution time is negatively affected. To decrease the number of new binary variables added throughout the process of approximation, Li et al. [70] proposed a piecewise linearization technique that contains a logarithmic number of variables with binary nature in \( n \). Consider the same continuous function \( f(x) \) expressed above, where \( x \) is assumed to be within the interval \([a_0, a_n]\) with \( n + 1 \) breakpoints. Let \( k \) be a positive integer number that can be represented using the following form:

\[
k = \sum_{i=0}^{h-1} 2^i \cdot y_i,
\]

(129)

where \( y_i \) are binary variables, \( n - 1 \) denotes the upper bound of \( k \), and \( h \) is set to \( \lfloor \log_2(n + 1) \rfloor \).

Consider a set \( A(k) \subseteq \{0, 1, \ldots, h - 1\} \) such that

\[
k = \sum_{i \in A(k)} 2^i.
\]

(130)

For example, \( A(0) = \emptyset \), \( A(3) = \{0, 1\} \), and \( A(5) = \{0, 2\} \). Denote \( \|A(k)\| \) to be the amount of elements in \( A(k) \). For example, \( \|A(0)\| = 0 \), and \( \|A(3)\| = 2 \). Li et al. [70] proposed the following expressions for approximation of a univariate non-linear function:

\[
L(f(x)) = \sum_{k=0}^{n-1} (f(a_k) - s_k(a_k - a_0)) \cdot r_k + \sum_{k=0}^{n-1} s_k \cdot w_k,
\]

(131)

s.t.

\[
x \geq \sum_{k=0}^{n-1} r_k \cdot a_k,
\]

(132)

\[
x \leq \sum_{k=0}^{n-1} r_k \cdot a_{k+1},
\]

(133)

\[
s_k = \frac{f(a_{k+1}) - f(a_k)}{a_{k+1} - a_k}, \quad \forall k \in \{0, \ldots, n - 1\},
\]

(134)

\[
\sum_{k=0}^{n-1} r_k = 1,
\]

(135)

\[
\sum_{k=0}^{n-1} r_k \cdot \|A(k)\| + \sum_{i=0}^{h-1} z_i = 0,
\]

(136)

\[
z_i \geq -y_i, \quad \forall i \in \{0, \ldots, h - 1\},
\]

(137)

\[
z_i \leq y_i, \quad \forall i \in \{0, \ldots, h - 1\},
\]

(138)

\[
z_i \geq \sum_{k=0}^{n-1} r_k \cdot c_{ki} - (1 - y_i), \quad \forall i \in \{0, \ldots, h - 1\},
\]

(139)

\[
z_i \leq \sum_{k=0}^{n-1} r_k \cdot c_{ki} + (1 - y_i), \quad \forall i \in \{0, \ldots, h - 1\},
\]

(140)

\[
\sum_{k=0}^{n-1} w_k = x - a_0,
\]

(141)

\[
\sum_{k=0}^{n-1} w_k \cdot \|A(k)\| + \sum_{i=0}^{h-1} v_i = 0,
\]

(142)
proposed by Li et al. [70] uses fewer variables of binary nature, Vielma et al. [71] showed that this approach is not a theoretically and computationally superior representation method.

Method 5

To approximate the non-linear functions of a variable, Vielma and Nemhauser [72] developed a new expression that needs fewer variables and constraints than previous representation methods. Their formulation requires a logarithmic number of binary variables and constraint sets in expressing a piecewise approximating function as follows. Let \( i \in I = \{0, 1, \ldots, n\} \) and \( B(i) = (y_1, y_2, \ldots, y_h) \), where \( h = \lceil \log_2 n \rceil \), \( y_k \in \{0, 1\} \) for \( k \in \{1, \ldots, h\} \). Let \( B : \{0, 1, \ldots, n\} \to \{0, 1\}^h \) denote the injective function, where the vectors \( B(i) \) and \( B(i + 1) \) differ in at most one component for all \( i \in \{0, \ldots, n - 1\} \) and \( B(0) = B(1) \). Let \( S^+(k) \) be a set of \( i \) where \( y_k = 1 \) in both \( B(i) \) and \( B(i + 1) \) for \( i \in \{0, \ldots, n - 1\} \) or just in \( B(i) \) for \( i \in \{0, n\} \) \( (S^+(k) = \{i \mid B(i) \text{ and } B(i + 1) \forall i \in \{1, \ldots, n - 1\} : y_k = 1 \} \cup \{i \mid B(i) \forall i \in \{0, n\} : y_k = 1 \}) \). Let \( S^-(k) \) be a set of \( i \) where \( y_k = 0 \) in both \( B(i) \) and \( B(i + 1) \) for \( i \in \{0, \ldots, n - 1\} \) or just in \( B(i) \) for \( i \in \{0, n\} \) \( (S^-(k) = \{i \mid B(i) \text{ and } B(i + 1) \forall i \in \{1, \ldots, n - 1\} : y_k = 0 \} \cup \{i \mid B(i) \forall i \in \{0, n\} : y_k = 0 \}) \). The piecewise linear function of \( f(x) \) with \( n + 1 \) breakpoints where \( a_0 < a_1 < \ldots < a_n \) can be represented as:

\[
L(f(x)) = \sum_{i=0}^{n} f(a_i) \cdot \lambda_i \tag{150}
\]

s.t.

\[
x = \sum_{i=0}^{n} a_i \cdot \lambda_i \tag{151}
\]

\[
\sum_{i=0}^{n} \lambda_i = 1, \tag{152}
\]

\[
\sum_{i \in S^+(k)} \lambda_i \leq y_k, \quad \forall k \in \{1, \ldots, h\}, \tag{153}
\]

\[
\sum_{i \in S^-(k)} \lambda_i \leq 1 - y_k, \quad \forall k \in \{1, \ldots, h\}, \tag{154}
\]

\[
y_k \in \{0, 1\}, \quad \forall k \in \{1, \ldots, h\}, \tag{155}
\]

\[
\lambda_i \in \mathbb{R}^+, \quad \forall i \in \{0, \ldots, n\}. \tag{156}
\]

To form a piecewise linear programming function with \( n \) pieces, this method applies \( \lceil \log_2 n \rceil \) variables of binary nature, \( n + 1 \) variables of continuous nature, and \( 2 + 2 \lceil \log_2 n \rceil \) constraint sets. Method 5 induces a piecewise linear function with an independent branch-
ing scheme of logarithm depth and constructs a tighter convex estimator, making fewer breakpoints to meet the feasibility and optimality tolerance. Experimental results from the literature \cite{71,73} show that this method provides a significant computational advantage in the linearization process and outperforms other methods, especially when more breakpoints are used.

### 3.1.2. PLA-Based Algorithms

Establishing the PLA generates its own optimization problem. To properly fit a curve, the most accurate PLA uses an infinite amount of segments. The complexity associated with this procedure becomes similar to the initial non-linear model. In solving the new decision problem, the main goal is to determine the best approximation with the least number of linear pieces. There are a number of commercial solvers, such as MINOS and CONOPT used within the General Algebraic Modeling System (GAMS), that can solve these problems \cite{74}. There are also several optimal search algorithms using PLAs to solve complicated models with both concave and convex objective functions \cite{75–78}. The most prominent algorithms developed based on PLA techniques include the following:

- **Approximating Planar Curves**

  PLAs are not just restricted to two-dimensional cases but can be utilized to fit multi-dimensional planes and curves. Williams \cite{79} presented the first algorithm that efficiently fits flat curves by using the required number of linear vectors. The proposed methodology of approximating the fitting of straight lines to a plane is based on the geometric analysis of the curvature of the plane, with subsequent geometrically accurate and numerically stable calculations.

- **Single Pass PLA Algorithm**

  Gritzali and Papakonstantinou \cite{80} devised an algorithm for finding different parts of a formulated waveform function and identifying a number of peak points. The points marked as peaks are those where the derivative of the function equals zero. This algorithm starts with a maximum point plotted on the piecewise curve and develops the PLA into the function in order that all points on the waveform have the same difference with respect to the piecewise approximation. Such an algorithm can obviously be beneficial in real-time applications like an electrocardiogram readout where peaks are important.

- **Branch and Refine**

  The branching and refining algorithm is based on the well-known PLA techniques. This is an effective way to find a global optimum for a non-linear problem. The algorithm applies PLAs to determine global lower bounds for mixed-integer non-linear optimization models, and from there, the upper bounds of the problem are provided by the feasible solutions. As the amount of iterations increases, the amount of segments increases, and the algorithm moves towards the global solution. For more information regarding the branching and refining algorithm, the interested readers can refer to Leyffer et al. \cite{81} and Gong and You \cite{82}.

- **PLAs for Accuracy**

  The algorithm designed by Nishikawa \cite{83} estimates the global and local asymptotic $L_2$ error as a piecewise continuous linear approximation in a manner that the target error on the curve can be achievable. The latter task can be accomplished via the local error analysis, which is followed by using key terms for the approximation. Some numerical tests are then performed to verify that the error is around $L_2$.

### 3.2. Log-Linearization via Taylor Series Approximation

There exist many different types of non-linear problems (e.g., dynamic stochastic general equilibrium), for which there exist no closed-form solutions and solving these problems is challenging. The log-linearization approach can be applied for such cases. When using the log-linearization approach, it is necessary to approximate the non-linear
equations characterizing the equilibrium with log-linear ones. The strategy is to first take
the natural logs of the non-linear equations and then use the first-order Taylor approximation
around the steady-state to replace the logged difference equations with linear approximations in the log-deviations of the variables. There are different ways to perform
log-linearization [84–86]. One of the main theories is to apply the Taylor Series expansion
as suggested by Griffith and Stewart [87]. Taylor’s theorem indicates that the first-order
approximation of an arbitrary mathematical function \( f(x) \) centered at \( x = x^* \) (differentiable
n times at some point \( x^* \)) can be represented as follows:

\[
f(x) \approx f(x^*) + f'(x^*) \cdot (x - x^*). \tag{157}
\]

For example, the function \( f(x) = \ln(1 + x) \) can be approximated at \( x = 2 \) by the
first-order Taylor polynomial as follows:

\[
f(x) \approx \ln 3 + \frac{1}{3} \cdot (x - 2) = 0.43195 + 0.3333 \cdot x. \tag{158}
\]

The first-order Taylor approximations can also be used to convert equations with more
than one endogenous variable to a log-deviation form. The first-order Taylor polynomial of
the function \( f(x, y) \) at the steady-state values \( x = x^* \) and \( y = y^* \) gives

\[
f(x, y) \approx f(x^*, y^*) + f_x'(x^*, y^*) \cdot (x - x^*) + f_y'(x^*, y^*) \cdot (y - y^*). \tag{159}
\]

This methodology could be used to log-linearize equations and take the log-deviation
around the steady state-value. Log-linearization means around a steady state. The log-deviation of the variable \( x \) from its steady state \( x^* \) is defined as

\[
x = \ln x - \ln x^*. \tag{160}
\]

The right-hand side of Equation (160) can be rewritten as

\[
\ln \left( \frac{x}{x^*} \right) = \ln \left( 1 + \frac{x - x^*}{x^*} \right). \tag{161}
\]

Using the first-order Taylor polynomial mentioned in Equation (157), the log expres-
sion can be approximated at the steady state \( x = x^* \) as

\[
\ln \left( 1 + \frac{x - x^*}{x^*} \right) \approx \ln 1 + \frac{1}{x^*} \cdot (x - x^*). \tag{162}
\]

Thus, we get log-deviation of \( x \) about \( x^* \) as

\[
\tilde{x} = \frac{x - x^*}{x^*}. \tag{163}
\]

Consider an example of the Cobb-Douglas production function \( y_t = a_t \cdot k_t^n \cdot n_t^{1-n} \) and
then take a log of the function:

\[
\ln y_t = \ln a_t + \alpha \cdot \ln k_t + (1 - \alpha) \cdot \ln n_t. \tag{164}
\]

Using Taylor Series expansion is the next step, we take the first order approximation
as follows:

\[
\ln y^* + \frac{1}{2} \cdot (y_t - y^*)
= \ln a^* + \frac{1}{2} \cdot (a_t - a^*) + \alpha \cdot \ln k^* + \frac{1}{2} \cdot (k_t - k^*)
+ (1 - \alpha) \cdot \ln n^* + \frac{1}{2(1-n)} \cdot (n_t - n^*). \tag{165}
\]

Since \( \ln y^* = \ln a^* + \alpha \cdot \ln k^* + (1 - \alpha) \cdot \ln n^* \), we can cancel out the relevant parts of
the approximation, which will result in the following expression:
\[
\frac{1}{y^*}(y_t - y^*) = \frac{1}{a^*}(a_t - a^*) + \frac{\alpha}{k^*}(k_t - k^*) + \frac{1 - \alpha}{n^*}(n_t - n^*). \tag{166}
\]

For notational ease, the equations are defined as the percentage deviation about the steady-state. Thus, applying log-deviation stated in Equation (163) to this approximation leads to

\[
\tilde{y}_t = \tilde{a}_t + \alpha \cdot \tilde{k}_t + (1 - \alpha) \cdot \tilde{n}_t. \tag{167}
\]

The method of taking logs and then subtracting the log terms from the steady-state equation is very convenient. However, it does not always work. It is only useful for multiplicative equations or when the log removes exponents and converts multiplication into addition to significantly simplify the equation. However, the method of taking logs and then subtracting the log terms from the steady-state equation should not be used for the equations that involve expectation terms, even when the equation is multiplicative \[85\]. This is because taking the expectation of a logarithmic term is not the same as taking the log from the expectation term.

4. Conclusions

In this study, a comprehensive and holistic review of transformation and linearization techniques was provided to deal with non-linear terms in optimization models. The following groups of operations research techniques for solving optimization problems with non-linear terms were analyzed: (i) transformations in which the non-linear equations or functions are replaced by an exact equivalent linear programming (LP) formulation to create valid inequalities; and (ii) linear approximations which find the equivalent of a non-linear function with the least deviation around the point of interest or separate straight-line segments. The existing transformation approaches for different scenarios were considered, including the multiplication of binary variables, multiplication of binary and continuous variables, multiplication of continuous variables, maximum/minimum operators, absolute value function, floor and ceiling functions, square root function, and multiple breakpoint function. As for linear approximations, the present survey provided a detailed review of piecewise approximating functions and log-linearization via Taylor series approximation.

The main advantages and disadvantages of using common transformation and linearization techniques were investigated as a part of this survey as well. Along with a review of the existing methods, this study proposed a new technique for linearizing the square root terms by means of transformation to obtain tight LP relaxations. Furthermore, a new approach was presented to incorporate quadratic integers into LP mathematical formulations. The aforementioned contributions are expected to enhance the solvability of optimization models with non-linear terms and, ultimately, facilitate decision making. Furthermore, the information presented in this survey study can be used by scientists and practitioners, who often work with optimization models that have non-linear terms, and selection of the appropriate transformation and linearization techniques. In conclusion, a detailed review of the relevant literature confirms that transformation methods were found to be efficient, as they ensure model feasibility and allow reducing the computation time.

The scope of future research for this study can focus more on additional techniques that can be used to decrease computational time even further after transformation of the original optimization model into its linear form (e.g., Lagrangian relaxation, Benders decomposition). Furthermore, a number of iterative optimization algorithms that directly rely on linearization techniques have been applied in the literature for different decision problems \[88–90\]. Another interesting research direction would be the investigation of computational complexity changes due to the deployment of linearization techniques in such algorithms. Moreover, future research can compare the computational performance of various PLA techniques in more detail for different scenarios. Last but not least, the future research can concentrate more on different domains where transformation and linearization techniques have been applied the most to discover additional tendencies and potential implementation challenges.
Author Contributions: Conceptualization: M.A. and S.M.J.M.A.-e.-h.; methodology: M.A. and S.M.J.M.A.-e.-h.; validation: M.A. and M.A.D.; formal analysis: M.A.; investigation: M.A.; writing—original draft preparation: M.A., A.M.F.-F. and M.A.D.; writing—review and editing: M.A., A.M.F.-F. and M.A.D.; supervision: M.A.; project administration: M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: This is a survey study and all the data are available within the main body of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proof of Theorem 1

Every integer number can be described in powers of 2 [91]. In other words, every positive integer number \( N \) lower than or equal to \( 2^n - 1 \) can be expressed as the sum of powers of 2 and \( n \) binary variables \( y_i \) as follows:

\[
N = 2^0 y_0 + 2^1 y_1 + \ldots + 2^{n-1} y_{n-1},
\]

(A1)

\[
y_i \in \{0,1\}, \quad \forall i \in \{0,\ldots,n-1\}.
\]

(A2)

For example, consider integer number \( N = 71,307 \). In this case, \( n \) is equal to 17 \((71,307 \leq 2^{17} - 1)\), and the binary equivalent raised to the power of 2 is

\[
71,307 = 2^0 y_0 + 2^1 y_1 + \ldots + 2^{16} y_{16}.
\]

(A3)

where \( y_i = 1 \) for \( i \in \{0, 1, 3, 7, 9, 10, 12, 16\} \) and other binary variables are zero.

Now, the above lemma can be extended to the case that the least upper bound is within \( 2^n - 1 < u < 2^{n+1} - 1 \). In this case, the integer number \( N \) with an upper bound of \( u \) can be written in the form

\[
N = 2^0 y_0 + 2^1 y_1 + \ldots + 2^{n-1} y_{n-1} + (u - 2^n + 1) y_u.
\]

(A4)

It should be pointed out that, as \( 2^n - 1 < u < 2^{n+1} - 1 \), \( n \) can be calculated by using \( \lfloor \log_2(u + 1) \rfloor \). For \( N \) less than or equal to \( 2^n - 1 \), it has been proven that the binary equivalent of \( \sum_{i=0}^{n-1} 2^i y_i \) can be used. Otherwise, for \( N \) greater than \( 2^n - 1 \), we only need to show that the binary equivalent can be made with the second component of Equation (A4), i.e., \((u - 2^n + 1) y_u\). The maximum value of \( \sum_{i=0}^{n-1} 2^i y_i \) is \( 2^n - 1 \), when all \( y_i \) take 1. As initially assumed \( u < 2^{n+1} - 1 \). Now, if we subtract \( 2^n - 1 \) from both sides of this equation, we obtain:

\[
u - (2^n - 1) < 2^{n+1} - 1 - (2^n - 1) = 2^n.
\]

(A5)

Since \( N \) is an integer number, Equation (A5) can be rewritten as follows:

\[
u - 2^n + 1 \leq 2^n - 1.
\]

(A6)

Therefore, it can be concluded that the second component of Equation (A4), i.e., \((u - 2^n + 1) y_u\), is always less than or equal to the maximum value of the first part, i.e., \( \sum_{i=0}^{n-1} 2^i y_i \). Assuming that \( N \) is greater than \( 2^n - 1 \), we only need to prove that the first part \( \sum_{i=0}^{n-1} 2^i y_i \) has the ability to produce the remaining amount, binary equivalent of \( N - (u - 2^n + 1) \). As \( N \geq 2^n - 1 \) and \( u - 2^n + 1 \leq 2^n - 1 \), then

\[
N - (u - 2^n + 1) \geq 0.
\]

(A7)
If we subtract \( u - 2^n + 1 \) from both sides of the initial condition \( N \leq u \), then
\[
N - (u - 2^n + 1) \leq 2^n - 1.
\] (A8)

Since \( 0 \leq \sum_{i=0}^{n-1} 2^i y_i \leq 2^n - 1 \), the first part can convert the remaining \( N - (u - 2^n + 1) \) into the binary equivalent. □

**Appendix B. Proof of Theorem 2**

In Theorem 1, we proved that any positive integer \( w \) with a limit of \( u \) can be converted into the binary equivalent. By rising the two sides of Equation (57) to the power of two, we attain the following binomial term:
\[
w^2 = \left( \sum_{i=0}^{w-1} 2^i y_i + (u - 2^n + 1) y_n \right)^2.
\] (A9)

By factorizing the right side of the above equation, we can rewrite it as follows:
\[
w^2 = \sum_{i=0}^{n-1} 2^{2i} y_i^2 + \sum_{i=0}^{n-2} \sum_{j=i}^{n-1} 2^{i+j+1} y_i y_j + (u - 2^n + 1) \sum_{i=0}^{n-1} 2^{i+1} y_i y_n
\]
\[+ (u - 2^n + 1)^2 y_n^2. \] (A10)

As mentioned in Section 2.1, since \( y_i \) is a binary variable, we can ignore the power of 2 and replace \( y_i^2 \) with \( y_i \). To linearize the term \( y_i y_j \), we replace it with a new binary variable \( z_{ij} (z_{ij} := y_i y_j) \) and add the following constraints:
\[
z_{ij} \leq y_i, \quad \forall i, j \in \{0, \ldots, n\}, \quad \text{(A11)}
\]
\[
z_{ij} \leq y_j, \quad \forall i, j \in \{0, \ldots, n\}, \quad \text{(A12)}
\]
\[
z_{ij} \geq y_i + y_j - 1, \quad \forall i, j \in \{0, \ldots, n\}, \quad \text{(A13)}
\]
\[
z_{ij} \in \{0, 1\}, \quad \forall i, j \in \{0, \ldots, n\}. \quad \text{(A14)}
\]

Let \( \beta = u - 2^n + 1 \). Then, Equation (A10) can be expressed as follows:
\[
w^2 = \sum_{i=0}^{n-1} 2^{2i} y_i + \sum_{i=0}^{n-2} \sum_{j>i}^{n-1} 2^{i+j+1} z_{ij} + \beta \sum_{i=0}^{n-1} 2^{i+1} z_{in} + \beta^2 y_n. \] (A15)

This completes the proof of Theorem 2. □

**Appendix C. Linearization of Quadratic Integers for \( n + 1 \)**

Extending this lemma for a given \( n + 1 \) implies that
\[
w_{n+1}^2 = \left( 2^0 y_0 + 2^1 y_1 + 2^2 y_2 + \ldots + 2^{n-1} y_{n-1} + 2^n y_n + \beta y_{n+1} \right)^2. \] (A16)

If we add \( \beta y_n - \beta y_{n+1} \) to the right of the above equation, then:
\[
w_{n+1}^2 = \left( 2^0 y_0 + 2^1 y_1 + 2^2 y_2 + \ldots + 2^{n-1} y_{n-1} + 2^n y_n + \beta y_{n+1} + \beta y_n - \beta y_{n+1} \right)^2. \] (A17)

According to Theorem 1, this equation can be rewritten as follows:
\[
w_{n+1}^2 = \left( w_n + 2^n y_n + \beta (y_{n+1} - y_n) \right)^2. \] (A18)

After factorization, we see that
\[
w_{n+1}^2 = w_n^2 + 2^{n+1} w_n y_n - 2 \beta w_n y_n + 2^n y_n^2 - 2^{n+1} \beta y_{n+1}^2 + \beta^2 y_n^2 
+ 2 \beta w_n y_n - 2^{n+1} \beta y_{n+1} + \beta^2 y_{n+1}. \] (A19)
According to the technique described in Section 2.1, we replace terms $y_n^2$ and $y_n \cdot y_{n+1}$ with variables $y_n$ and $z_{n,n+1}$, respectively. Then

\[
 w_{n+1}^2 = w_n^2 + w_n \cdot y_n \cdot (2^{n+1} - 2\beta) + 2\beta \cdot w_n \cdot y_{n+1} + (\beta^2 - 2^{n+1} \beta + 2^{2n}) \cdot y_n + \\
 \beta^2 \cdot y_{n+1} + z_{n,n+1} \cdot (2^{n+1} - 2\beta^2),
\]

where

\[
 w_n \cdot y_n = \left( \sum_{i=0}^{n-1} 2^i \cdot y_i + \beta \cdot y_n \right) \cdot y_n = \sum_{i=0}^{n-1} 2^i \cdot z_{ih} + \beta \cdot y_n,
\]

\[
 w_n \cdot y_{n+1} = \left( \sum_{i=0}^{n-1} 2^i \cdot y_i + \beta \cdot y_n \right) \cdot y_{n+1} = \sum_{i=0}^{n-1} 2^i \cdot z_{n,n+1} + \beta \cdot z_{n,n+1}.
\]

Finally, by replacing $w_n$ with $\sum_{i=0}^{n-1} 2^i \cdot y_i + \beta \cdot y_n$ and simplifying the equation, we obtain the linearized form for $n + 1$ shown below:

\[
 w_{n+1}^2 = \sum_{i=0}^{n-1} 2^i \cdot y_i + 2^{2n} \cdot y_n + \left( \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 2^{i+j+1} \cdot z_{ij} \right) + \beta^2 \cdot y_{n+1} + \\
 + \left( \sum_{i=0}^{n-1} 2^{i+1} \cdot z_{i,n+1} + 2^{n+1} \cdot z_{n,n+1} \right) + \beta^2 \cdot y_{n+1}.
\]

References
1. Negrello, C.; Gosselet, P.; Rey, C. Nonlinearly Preconditioned FETI Solver for Substructured Formulations of Nonlinear Problems. Mathematics 2021, 9, 3165. [CrossRef]


89. Dulebenets, M.A. The green vessel scheduling problem with transit time requirements in a liner shipping route with Emission Control Areas. Alex. Eng. J. 2018, 57, 331–342. [CrossRef]
