Palestine Journal of Mathematics

Vol 14 (Special Issue IV), (2025) , 187–195

Approximating Solution for Posynomial Geometric Programming Problems using Interval Valued Function

Harpreet Singh and Amanpreet Singh

MSC 2010 Classifications: Primary 33C20; Secondary 33C65.

Keywords and phrases: Posynomial Geometric Programming, Interval-valued function, Karush-Kuhn Tucker conditions.

Abstract Geometric programming involving posynomials presents significant complexity when the coefficients are not represented as a single constant instead vary within a specific range. Numerous techniques have been followed to obtain the optimal solution to these kinds of problems. This paper focuses on PGPP's having single objective function in the case when interval coefficients not only occur in the objective function but also on both the sides of the constraints. To achieve the optimality of the solution, interval coefficients are converted to a single representative value by employing interval-valued function thus reducing them to standard nonlinear programming problem. Afterwards, Karush-Kuhn-Tucker conditions and Taylor's series expansion for multivariable taken upto first order leads us to derive the optimal solution. The function codes have been implemented in Python and performed using Google Colab.

1 Introduction

The fundamental objective of geometric programming is to obtain optimal solution for single-objective and multi-objective geometric programming problems involving posynomial terms. The foundation of theory of geometric programming was laid down by Duffin, Peterson and Zener in the year 1967. Various solution techniques had been developed by Beightler et al. [1], Avriel et al. [2], Duffin et al. [3] for obtaining the solution of GPP's.

Since past decade, geometric programming is gaining keen attention of the researchers. Liu [[4], [5]], Ojha et al. [6], Mahapatra [7] proposed solution procedures for various complex forms of the GPP when not only the coefficients of the posynomials terms but also the exponents of the variables were given as multiple parameters. Ojha and Biswal [8] formulated a solution procedure of multi-objective geometric programming problems using weighted mean method. Mousavi and Saraj [9] implemented parametric approach proposed by Mahapatra and Mandal [7] on multi-objective geometric programming problems (MOGPP) with interval coefficients. Das & Roy [10], compared solutions of multi-objective Gravel box problem obtained by weighted-sum and weighted product methods with weighted min-max method. Ojha, Ota [11] implemented a hybrid approach to solve MOGPP by combining ϵ-constraint method with that of weighted mean method. A dynamic approach was introduced by Oz et al. [12], where KKT conditions were used to minimize the weighted objective function thus reducing MOGPP to a linear programming problem using Taylor's series expansion. Singh et al. [13] proposed a blended method in which the numerical approach proposed by Erzoy et al. for MOGPP was implemented on single objective PGPP with interval coefficients by initially transforming interval coefficient into single value using interval valued function.

Abdullah and Rasul [14] solved multi-objective GPP by initially transforming the MOGPP to a SGPP using two different algorithms. The transformed SGPP was then solved using dual programming. Amuji et al. [15] used a technique to refine linear programming problem using geometric programming resulting in improved solution. Modal et al. [16] aimed to present solution procedures for geometric programming problems characterized by triangular and trapezoidal uncertainty distributions. This paper presents numerical examples to demonstrate the efficacy of the procedures and algorithms.

In section 2 an algorithm to solve the problem is proposed and illustrated with the help of an

example given in section 4 followed by a solution table. The results so obtained are concluded in section 5.

2 Basic Terminology

2.1 Standard form of Posynomial Geometric Programming Problem

The Standard posynomial geometric programming problem when the coefficients are single values is stated as:

$$\text{Min } f_0(x) = \sum_{i=1}^{n} p_i \prod_{j=i}^{m} x_j^{q_{ij}}$$

$$\text{Subject to} \quad f_k(x) = \sum_{i=1}^{l_k} r_{ik} \prod_{j=i}^{m} x_j^{s_{ij}} \leq 1, \quad k = 1, 2, \ldots, t$$

$$x_j > 0, \quad j = 1, 2, \ldots\ldots\ldots, m$$

where the exponents q_{ij}, s_{ij} assumed to be arbitrary real numbers and the coefficients p_i, r_{ik} can take real values.

2.2 Single-Objective Posynomial Geometric Programming Problem when coefficients are intervals

The general structure of single-objective posynomial geometric programming problem when coefficients are intervals is defined as:

$$\text{Min } f_0(x) = \sum_{i=1}^{n} [p_i, q_i] \prod_{j=i}^{m} x_j^{q_{ij}}$$

$$\text{Subject to } f_k(x) = \sum_{i=1}^{l_k} [p_{ik}, q_{ik}] \prod_{j=i}^{m} x_j^{s_{ij}} \leq [b_{ik}, c_{ik}], \quad k = 1, 2, \ldots\ldots, t$$

$$\text{with } x_j > 0, \quad j = 1, 2, \ldots\ldots\ldots, m$$

where the exponents q_{ij}, S_{ij} assumed to be arbitrary real numbers and the constants $p_i, q_i, p_{ik}, q_{ik}, b_{ik}, c_{ik}$ can take positive real values.

2.3 Karush-Kuhn Tucker conditions

For general mathematical programming problem:

$$\text{Min } f_0(x)$$

$$\begin{aligned} \text{s.t} \qquad f_k(x) &\leq 0, \quad k = 1, 2, \ldots, t \\ x_i &> 0, \quad i = 1, 2, \ldots., t \end{aligned}$$

KKT conditions are given as:

$$\frac{\partial f_0(x)}{\partial x_i} + \sum_{k=1}^{t} \lambda_k \frac{\partial f_k(x)}{\partial x_i} = 0, \quad i = 1, 2 \ldots, t$$

$$\begin{aligned} f_k(x) &\leq 0, \quad k = 1, 2, \ldots\ , t \\ \lambda_k f_k(x) &= 0, \quad k = 1, 2, \ldots, t \\ x_i &> 0, \quad i = 1, 2, \ldots, t \end{aligned}$$

2.4 Interval-valued function

Let $[m, n]$ be any interval with $m > 0, n > 0$. Then the interval valued function $f(q)$ corresponding to the interval $[m, n]$ can be stated as:

$$f(q) = m^{1-q} n^q, \qquad 0 \le q \le 1$$

3 Algorithm to solve the problem

Step I Firstly, select a problem with coefficients as intervals and use the interval-valued function definition [2.3] to transform the interval into a single number.

Step II Use Langrangian theorem to construct new objective function and apply KKT conditions to formulate the constraints for the corresponding objective function.

Step III Use Taylor's series expansion about any random feasible point to transform the system of non-linear exression obtained in Step II to linear one.

Step IV Solve the LPP obtained in Step III using Python programming. Use Google Colab to execute the programs.

4 Numerical Example

The proposed structure of the algorithm is demonstrated with the help of an example. In this example, a complex form of above said type is considered in which the coefficients of the posynomial term in the objective function and the constraints and also the constant terms are given as intervals. The below problem was considered by Liu [4] and Mahapatra & Mandal [7].

Problem:

$$\text{Min } f(x) = (2,3)\ x_1^2 x_2^{-1} x_3 x_4^{-1} + (4, 4.2)\ x_1^{-3} x_2^2 x_3^{-2}$$

Subject to

$$(3, 3.6)\ x_1^3 x_3 + x_1^{-1} x_3^{-1} \le (2, 4)$$
$$x_2^{-1} x_3 x_4 + (2, 2.8)\ x_1^2 x_2 x_4 \le 1$$
$$x_1, x_2, x_3, x_4 > 0$$

Step I With the help of interval valued function, the intervals of the problem are transformed to the following form:

$$\text{Min}[f(x)]_q = 2^{1-q} 3^q x_1^2 x_2^{-1} x_3 x_4^{-1} + 4^{1-q} (4.2)^q x_1^{-3} x_2^2 x_3^{-2}$$

Subject to

$$3^{1-q} (3.6)^q\ x_1^3 x_3 + x_1^{-1} x_3^{-1} \le 2^q 4^{1-q},$$
$$x_2^{-1} x_3 x_4 + 2^{1-q} (2.8)^q\ x_1^2 x_2 x_4 \le 1$$
$$x_1, x_2, x_3, x_4 > 0, \qquad 0 \le q \le 1$$

Or

$$(0.75)^{1-q} (1.8)^q\ x_1^3 x_3 + (0.5)^q (0.25)^{1-q}\ x_1^{-1} x_3^{-1} \le 1$$
$$x_2^{-1} x_3 x_4 + 2^{1-q} (2.8)^q\ x_1^2 x_2 x_4 \le 1$$
$$x_1, x_2, x_3, x_4 > 0 \text{ where } 0 \le q \le 1$$

With the use of slack variables, the problem is rewritten as:

$$\text{Min}[f(x)]_q = 2^{1-q} 3^q x_1^2 x_2^{-1} x_3 x_4^{-1} + 4^{1-q} (4.2)^q x_1^{-3} x_2^2 x_3^{-2}$$

Subject to

$$(0.75)^{1-q} (1.8)^q\ x_1^3 x_3 + (0.5)^q (0.25)^{1-q}\ x_1^{-1} x_3^{-1} + \gamma_1^2 - 1 = 0$$
$$x_2^{-1} x_3 x_4 + 2^{1-q} (2.8)^q\ x_1^2 x_2 x_4 + \gamma_2^2 - 1 = 0$$
$$x_1, x_2, x_3, x_4 > 0, \qquad 0 \le q \le 1$$

Step II With the help of Langrange's multipliers (y_1, y_2) say, the objective function takes the form:

$$\begin{aligned}\text{Min}[f(x)]_{q,y_1,y_2} = 2^{1-q}3^q\ x_1^2x_2^{-1}x_3x_4^{-1}+4^{1-q}(4.2)^q\ x_1^{-3}x_2^2x_3^{-2}-y_1\left(1-(0.75)^{1-q}(1.8)^q\ x_1^3x_3-\right.\\ \left.(0.5)^q(0.25)^{1-q}\ x_1^{-1}x_3^{-1}-\gamma_1^2\right)-y_2\left(1-x_2^{-1}x_3x_4-2^{1-q}(2.8)^q\ x_1^2x_2x_4-\gamma_2^2\right)\end{aligned}$$

After applying KKT conditions, the above problem is reformulated as:

$$\text{Min}[f(x)]_{q,y_1,y_2} = 2^{1-q}3^q\ x_1^2x_2^{-1}x_3x_4^{-1} + 4^{1-q}(4.2)^q\ x_1^{-3}x_2^2x_3^{-2} \tag{4.1}$$

Subject to

$$\begin{aligned}2\times 2^{1-q}3^qx_1x_2^{-1}x_3x_4^{-1} + (-3)\times 4^{1-q}(4.2)^qx_1^{-4}x_2^2x_3^{-2} - y_1\left(-3\times(0.75)^{1-q}(1.8)^qx_1^2x_3\right.\\ \left.+(0.5)^q(0.25)^{1-q}x_1^{-2}x_3^{-1}\right)-y_2\left(-2\times 2^{1-q}(2.8)^qx_1x_2x_4\right)=0\end{aligned} \tag{4.2}$$

$$-2^{1-q}3^qx_1^2x_2^{-2}x_3x_4^{-1}+2\times 4^{1-q}(4.2)^qx_1^{-3}x_2x_3^{-2}-y_2\left(x_2^{-2}x_3x_4-2^{1-q}(2.8)^qx_1^2x_4\right)=0 \tag{4.3}$$

$$\begin{aligned}2^{1-q}3^qx_1^2x_2^{-1}x_4^{-1}+(-2)\times 4^{1-q}(4.2)^qx_1^{-3}x_2^2x_3^{-3}-y_1\left(-(0.75)^{1-q}(1.8)^qx_1^3\right.\\ \left.+(0.5)^q(0.25)^{1-q}x_1^{-1}x_3^{-2}\right)-y_2\left(-x_2^{-1}x_4\right)=0\end{aligned} \tag{4.4}$$

$$-2^{1-q}3^qx_1^2x_2^{-1}x_3x_4^{-2}-y_2\left(-x_2^{-1}x_3-2^{1-q}(2.8)^qx_1^2x_2\right)=0 \tag{4.5}$$

$$y_1\left(1-(0.75)^{1-q}(1.8)^qx_1^3x_3-(0.5)^q(0.25)^{1-q}x_1^{-1}x_3^{-1}\right)\geq 0 \tag{4.6}$$

$$y_2\left(1-x_2^{-1}x_3x_4-2^{1-q}(2.8)^qx_1^2x_2x_4\right)\geq 0 \tag{4.7}$$

Step III Further, using Taylor's series expansion upto first order about any random initial feasible point X^0, the objective function (4.1) and constraints (4.2) to (4.7) becomes:

$$\begin{aligned}\text{Min}[f(x)]_{q,y_1,y_2}\approx\left[2^{1-q}3^qx_1^2x_2^{-1}x_3x_4^{-1}+4^{1-q}(4.2)^qx_1^{-3}x_2^2x_3^{-2}\right]_{X^0}+\\ \left[2\times 2^{1-q}3^qx_1x_2^{-1}x_3x_4^{-1}+(-3)\times 4^{1-q}(4.2)^qx_1^{-4}x_2^2x_3^{-2}\right]_{X^0}(x_1-x_1^0)+\\ \left[-2^{1-q}3^qx_1^2x_2^{-2}x_3x_4^{-1}+2\times 4^{1-q}(4.2)^qx_1^{-3}x_2x_3^{-2}\right]_{X^0}(x_2-x_2^0)+\\ \left[2^{1-q}3^qx_1^2x_2^{-1}x_4^{-1}+(-2)\times 4^{1-q}(4.2)^qx_1^{-3}x_2^2x_3^{-3}\right]_{X^0}(x_3-x_3^0)+\\ \left[-2^{1-q}3^qx_1^2x_2^{-1}x_3x_4^{-2}\right]_{X^0}(x_4-x_4^0)\end{aligned} \tag{4.8}$$

Subject to

$$\Big[2\times 2^{1-q}3^q x_1x_2^{-1}x_3x_4^{-1}+(-3)\times 4^{1-q}(4.2)^q x_1^{-4}x_2^2x_3^{-2}-y_1\left(-3\times(0.75)^{1-q}(1.8)^q x_1^2x_3\right)$$
$$+(0.5)^q(0.25)^{1-q}x_1^{-2}x_3^{-1}-y_2\left(-2\times 2^{1-q}(2.8)^q x_1x_2x_4\right)\Big]_{X^0}+$$
$$\Big[2\times 2^{1-q}\cdot 3^q\, x_2^{-1}x_3x_4^{-1}+12\times 4^{1-q}(4.2)^q x_1^{-5}x_2^2x_3^{-2}-y_1\left(-6\times(0.75)^{1-q}(1.8)^q x_1x_3\right)$$
$$+(-2)\times(0.5)^q(0.25)^{1-q}x_1^{-3}x_3^{-1}-y_2\left(-2\times 2^{1-q}(2.8)^q x_2x_4\right)\Big]_{X^0}(x_1-x_1^0)+$$
$$\Big[-2\times 2^{1-q}\cdot 3^q\, x_1x_2^{-2}x_3x_4^{-1}-6\times 4^{1-q}(4.2)^q x_1^{-4}x_2x_3^{-2}-y_2\left(-2\times 2^{1-q}(2.8)^q x_1x_4\right)\Big]_{X^0}(x_2-x_2^0)+$$
$$\Big[2\times 2^{1-q}\cdot 3^q\, x_1x_2^{-1}x_4^{-1}+6\times 4^{1-q}(4.2)^q x_1^{-4}x_2^2x_3^{-3}-y_1\left(-3\times(0.75)^{1-q}(1.8)^q x_1^2\right)$$
$$-(0.5)^q(0.25)^{1-q}x_1^{-2}x_3^{-2}\Big]_{X^0}(x_3-x_3^0)+$$
$$\Big[-2\times 2^{1-q}\cdot 3^q\, x_1x_2^{-1}x_3x_4^{-2}-y_2\left(-2\times 2^{1-q}(2.8)^q x_1x_2\right)\Big]_{X^0}(x_4-x_4^0)-$$
$$\Big[y_1\left(-3\times(0.75)^{1-q}(1.8)^q x_1^2x_3+(0.5)^q(0.25)^{1-q}x_1^{-2}x_3^{-1}\right)\Big]_{X^0}(y_1-y_1^0)-$$
$$\left[(-2\times 2^{1-q}(2.8)^q x_1x_2x_4)\right]_{X^0}(y_2-y_2^0)=0 \quad (4.9)$$

$$\Big[-2^{1-q}3^q x_1x_2^{-2}x_3x_4^{-1}+2\times 4^{1-q}(4.2)^q x_1^{-3}x_2x_3^{-2}-y_2(-x_2^{-1}x_3x_4-2^{1-q}(2.8)^q x_1^2x_4)\Big]_{X^0}+$$
$$\Big[-2\times 2^{1-q}3^q x_1x_2^{-2}x_3x_4^{-1}-6\times 4^{1-q}(4.2)^q x_1^{-4}x_2x_3^{-2}+y_2\times 2\times 2^{1-q}(2.8)^q x_1x_4\Big]_{X^0}(x_1-x_1^0)+$$
$$\Big[2\times 2^{1-q}3^q x_1x_2^{-3}x_3x_4^{-1}+2\times 4^{1-q}(4.2)^q x_1^{-3}x_3^{-2}+y_2\times 2x_2^{-3}x_3x_4\Big]_{X^0}(x_2-x_2^0)+$$
$$\Big[-2^{1-q}3^q x_1x_2^2x_2^{-2}x_4^{-1}-4\times 4^{1-q}(4.2)^q x_1^{-3}x_2x_3^{-3}-y_2x_2^{-2}x_4\Big]_{X^0}(x_3-x_3^0)+$$
$$\Big[2^{1-q}3^q x_1x_2^{-2}x_3x_4^{-2}-y_2(x_2^{-2}x_3-2^{1-q}(2.8)^q x_1^2)\Big]_{X^0}(x_4-x_4^0)-$$
$$\Big[x_2^{-2}x_3x_4-2^{1-q}(2.8)^q x_1^2x_4\Big]_{X^0}(y_2-y_2^0)=0 \quad (4.10)$$

$$\Big[2^{1-q}\cdot 3^q\cdot x_1^2x_2^{-1}x_4^{-1}-2\cdot 4^{1-q}\cdot(4.2)^q\cdot x_1^{-3}x_2^2x_3^{-3}-$$
$$y_1\left(-(0.75)^{1-q}(1.8)^q x_1^3+(0.5)^q(0.25)^{1-q}x_1^{-1}x_3^{-2}\right)-y_2(-x_2^{-1}x_4)\Big]_{X^0}+$$
$$\Big[2\times 2^{1-q}\cdot 3^q\cdot x_1\cdot x_2^{-2}\cdot x_4^{-1}+6\times 4^{1-q}\cdot(4.2)^q\cdot x_1^{-4}\cdot x_2^2\cdot x_3^{-3}-$$
$$y_1\left(-3\times(0.75)^{1-q}\cdot(1.8)^q\cdot x_1^2-(0.5)^q\cdot(0.25)^{1-q}\cdot x_1^{-2}\cdot x_3^{-2}\right)\Big]_{X^0}(x_1-x_1^0)+$$
$$\Big[-2^{1-q}\cdot 3^q\cdot x_1^2x_2^{-2}x_4^{-1}-4\cdot 4^{1-q}\cdot(4.2)^q\cdot x_1^{-3}x_2x_3^{-3}-y_2\left(x_2^{-2}x_4\right)\Big]_{X^0}(x_2-x_2^0)+$$
$$\Big[6\times 4^{1-q}(4.2)^q\, x_1^{-3}x_2^2x_3^{-4}+2\times y_1\,(0.5)^q\,(0.25)^{1-q}\,x_1^{-1}x_3^{-3}\Big]_{X^0}(x_3-x_3^0)+$$
$$\Big[-2^{1-q}3^q x_1^2x_2^{-1}x_4^{-2}-y_2(-x_2^{-2})\Big]_{X^0}(x_4-x_4^0)+$$
$$\Big[(0.75)^{1-q}(1.8)^q x_1^3+(0.5)^q(0.25)^{1-q}x_1^{-1}x_3^{-2}\Big]_{X^0}(y_1-y_1^0)+\Big[x_2^{-1}x_4\Big]_{X^0}(y_2-y_2^0)=0 \quad (4.11)$$

$$\left[-2^{1-q}\cdot 3^q\cdot x_1^2x_2^{-1}x_3x_4^{-2}-y_2\left(-x_2^{-1}x_3-2^{1-q}(2.8)^q x_1^2x_2\right)\right]_{X^0}+$$
$$\left[-2\cdot 2^{1-q}\cdot 3^q\cdot x_1x_2^{-1}x_3x_4^{-2}+y_2\left(2\cdot 2^{1-q}(2.8)^q x_1x_2\right)\right]_{X^0}(x_1-x_1^0)+$$
$$\left[2^{1-q}\cdot 3^q\cdot x_1^2x_2^{-2}x_3x_4^{-2}-y_2\left(-x_2^{-2}x_3-2^{1-q}(2.8)^q x_1^2\right)\right]_{X^0}(x_2-x_2^0)+$$
$$\left[-2^{1-q}\cdot 3^q\cdot x_1^2x_2^{-1}x_4^{-2}+y_2x_2^{-1}\right]_{X^0}(x_3-x_3^0)+\left[2\cdot 2^{1-q}\cdot 3^q\cdot x_1^2x_2^{-1}x_3x_4^{-3}\right]_{X^0}(x_4-x_4^0)+$$
$$\left[x_2^{-1}x_3-2^{1-q}(2.8)^q x_1^2x_2\right]_{X^0}(y_2-y_2^0)=0 \quad (4.12)$$

$$\left[y_1\left(1-(0.75)^{1-q}\cdot(1.8)^q x_1^3x_3-(0.5)^q\cdot(0.25)^{1-q}x_1^{-1}x_3^{-1}\right)\right]_{X^0}+$$
$$\left[y_1\left(-3\cdot(0.75)^{1-q}\cdot(1.8)^q x_1^2x_3+(0.5)^q\cdot(0.25)^{1-q}x_1^{-2}x_3^{-1}\right)\right]_{X^0}(x_1-x_1^0)+$$
$$\left[y_1\left(-(0.75)^{1-q}\cdot(1.8)^q x_1^3+(0.5)^q\cdot(0.25)^{1-q}x_1^{-1}x_3^{-2}\right)\right]_{X^0}(x_3-x_3^0)+$$
$$\left[1-(0.75)^{1-q}\cdot(1.8)^q x_1^3x_3-(0.5)^q\cdot(0.25)^{1-q}x_1^{-1}x_3^{-1}\right]_{X^0}(y_1-y_1^0)\geq 0 \quad (4.13)$$

$$\left[y_2\left(1-x_2^{-1}x_3x_4-2^{1-q}\cdot(2.8)^q x_1^2x_2x_4\right)\right]_{X^0}+\left[y_2\left(-2\cdot 2^{1-q}\cdot(2.8)^q x_1x_2x_4\right)\right]_{X^0}(x_1-x_1^0)+$$
$$\left[y_2\left(x_2^{-2}x_3x_4-2\cdot 2^{1-q}\cdot(2.8)^q x_1x_2x_4\right)\right]_{X^0}(x_2-x_2^0)+\left[y_2\left(-x_2^{-1}x_4\right)\right]_{X^0}(x_3-x_3^0)+$$
$$\left[y_2\left(-x_2^{-1}x_3-2^{1-q}\cdot(2.8)^q x_1^2x_2\right)\right]_{X^0}(x_4-x_4^0)+$$
$$\left[1-x_2^{-1}x_3x_4-2^{1-q}\cdot(2.8)^q x_1^2x_2x_4\right]_{X^0}(y_2-y_2^0)\geq 0 \quad (4.14)$$

$$x_1,x_2,x_3,x_4>0 \text{ where } 0\leq q\leq 1$$

Step IV Let us consider the random feasible point X^0 to be:

$$x_1^0=0.5, x_2^0=0.5, x_3^0=1.6, x_4^0=0.2, y_1^0=0.7, y_2^0=1.1$$

Since $0\leq q\leq 1$, so by varying q in the given range, the corresponding optimal solutions can be obtained. We have obtained the solution for different q as shown in table 1 .

Corresponding to $q=0$ and initial feasible point X^0, the codes for the objective function (4.8) and the constraints (4.9) to (4.14) are written below.

For calculation purpose, while using web based Google colaboratory, x_1, x_2, x_3, x_4, y_1 and y_2 are represented by a_1, a_2, a_3, a_4, b_1 and b_2 and $x_1^0, x_2^0, x_3^0, x_4^0, y_1^0, y_2^0$ are written as x_1, x_2, x_3, x_4, y_1 and y_2 .

4.1 Python code for objective function (4.8)

```
a1 = symbols('a1')*
a2 = symbols('a2')
a3 = symbols('a3')
a4 = symbols('a4')
Min f=((pow(2,1-q)*pow(3,q)*pow(x1,2)*pow(x2,-1)*pow(x3,1)*pow(x4,-1))
+(pow(4,1-q)*pow(4.2,q)*pow(x1,-3)*pow(x2,2)*pow(x3,-2)))+(((2*pow(2,1-q)*
pow(3,q)*pow(x2,-1)*x1*x3*pow(x4,-1))-(3*pow(4,1-q)*pow(4.2,q)*pow(x1,-4)*
pow(x2,2)*pow(x3,-2)))*(a1-x1))+(((-1*pow(2,1-q)*pow(3,q)*pow(x2,-2)*
pow(x1,2)*x3*pow(x4,-1))+((2*pow(4,1-q)*pow(4.2,q))*x2*pow(x1,-3)*
pow(x3,-2)))*(a2-x2))+(((pow(2,1-q)*pow(3,q)*pow(x1,2)*pow(x2,-1)*
pow(x4,-1))-(2*pow(4,1-q)*pow(4.2,q)*pow(x1,-3)*pow(x2,2)*pow(x3,-3)))*
(a3-x3))-((pow(2,1-q)*pow(3,q)*pow(x1,2)*pow(x2,-1)*x3*pow(x4,-2))*(a4-x4))
smpl = simplify(minz)
smpl
```

After substituting the initial point, the above equation reduces to:
$1.325a1 - 3.5a2 + 1.09375a3 - 40.0a4 + 12.5$

4.2 The Codes for constraints (4.9) and the corresponding equation after substituting initial values are as follows:

```
Min C1=((2*pow(2,1-q)*pow(3,q)*pow(x1,1)*pow(x2,-1)*pow(x3,1)*pow(x4,-1))-
(3*pow(4,1-q)*pow(4.2,q)*pow(x1,-4)*pow(x2,2)*pow(x3,-2))-
(y1*((-1*pow(0.75,1-q)*pow(1.8,q)*3*pow(x1,2)*x3)+
(pow(0.5,q)*pow(0.25,1-q)*pow(x1,-2)*pow(x3,-1))))+(y2*pow(2,1-q)*
pow(2.8,q)*2*x1*x2*x4))+(((2*pow(2,1-q)*pow(3,q)*
pow(x2,-1)*x3*pow(x4,-1))+(12*pow(4,1-q)*pow(4.2,q)*
pow(x1,-5)*pow(x2,2)*pow(x3,-2))-(y1*((-1*pow(0.75,1-q)*
pow(1.8,q)*6*x1*x3)-(2*pow(0.5,q)*pow(0.25,1-q)*
pow(x1,-3)*pow(x3,-1))))+(y2*pow(2,1-q)*pow(2.8,q)*2*x2*x4))*
(a1-x1))+(((-2*pow(2,1-q)*pow(3,q)*pow(x2,-2)*pow(x1,1)*x3*
pow(x4,-1))-((6*pow(4,1-q)*pow(4.2,q))*x2*pow(x1,-4)*
pow(x3,-2))+(y2*pow(2,1-q)*pow(2.8,q)*2*x1*x4))*(a2-x2))+
(((pow(2,1-q)*pow(3,q)*2*pow(x1,1)*pow(x2,-1)*pow(x4,-1))+
(6*pow(4,1-q)*pow(4.2,q)*pow(x1,-4)*pow(x2,2)*pow(x3,-3))-
(y1*((-1*pow(0.75,1-q)*pow(1.8,q)*3*pow(x1,2))-(pow(0.5,q)*
pow(0.25,1-q)*pow(x1,-2)*pow(x3,-2)))))*(a3-x3))+
(((-2*pow(2,1-q)*pow(3,q)*pow(x1,1)*pow(x2,-1)*x3*pow(x4,-2))+
(y2*pow(2,1-q)*pow(2.8,q)*2*x1*x2))*(a4-x4))-
(((-1*pow(0.75,1-q)*pow(1.8,q)*3*pow(x1,2)*x3)+(pow(0.5,q)*
pow(0.25,1-q)*pow(x1,-2)*pow(x3,-1)))*(b1-y1))+((pow(2,1-q)*
pow(2.8,q)*2*x1*x2*x4)*(b2-y2))
smpl = simplify(minz)
smpl
```

$218.71a1 - 138.56a2 + 44.1046875a3 - 158.9a4 + 0.275b1 + 0.2b2 - 65.6125 = 0$
Similarly, using Python programming, we calculate the values of constraints (4.10) to (4.14) after substituting the initial values.
Thus the resulting LPP comes out to be (rounded off upto 4 decimal places):

$$\textbf{Min } F = 13.25a13.5a2 + 1.09375a340.0a4 + 12.5$$

Subject to

$$\begin{aligned}
218.71a_1 - 138.56a_2 + 44.1046875a_3 - 158.9a_4 + 0.275b_1 + 0.2b_2 - 65.6125 &= 0\\
-138.56a_1 + 94.632a_2 - 26.505a_3 + 73.51a_4 - 1.18b_2 + 46.17 &= 0\\
44.1046875a_1 - 26.505a_2 + 7.4951171875a_3 - 22.8a_4 - 0.1015625b_1 + 0.4b_2 - 15.13828125 &= 0\\
-158.9a_1 + 73.51a_2 - 22.8a_3 + 400.0a_4 + 3.45b_2 - 40.825 &= 0\\
-0.1925a_1 + 0.07109375a_3 + 0.5375b_1 - 0.0175 &\geq 0\\
-0.22a_1 + 1.298a_2 - 0.44a_3 - 3.795a_4 + 0.31b_2 + 0.9240 &\geq 0
\end{aligned}$$

The above LPP is again solved using Python based Google Colab to obtain the optimal solution as below:

4.3 The corresponding codes for the solution are given as:

```
# Create an object of a model
prob = LpProblem("Simple LP Problem", LpMinimize)
# Define the decision variables
x1 = LpVariable("x1", 0)
x2 = LpVariable("x2", 0)
x3 = LpVariable("x3", 0)
x4 = LpVariable("x4", 0)
y1= LpVariable("y1", 0)
y2= LpVariable("y2", 0)
# Define the objective function
prob += (13.25*x1) - (3.5*x2) + (1.0938*x3) - (40*x4) + (12.5)
# Define the constraints
```

```
prob += (218.71*x1) - (138.56*x2) + (44.1047*x3) - (158.9*x4) + (0.275*y1)
+ (0.2*y2) - (65.6125) == 0, "1st constraint"
prob += - (138.56*x1) + (94.632*x2) - (26.505*x3) + (73.51*x4) - (1.18*y2)
+ (46.17) == 0, "2nd constraint"
prob += (44.1047*x1) - (26.505*x2) + (7.4951*x3) - (22.8*x4) - (0.1016*y1)
+ (0.4*y2) - (15.1383) == 0, "3rd constraint"
prob += - (158.9*x1) + (73.51*x2) - (22.8*x3) + (400*x4) + (3.45*y2) -
(40.825) == 0, "4th constraint"
prob += - (0.1925*x1) + (0.0711*x3) + (0.5375*y1) - (0.0175) >= 0, "5th
constraint"
prob += - (0.22*x1) + (1.298*x2) - (0.44*x3) - (3.795*x4) + (0.31*y2) +
(0.924) >= 0, "6th constraint"
prob.solve()
1
# Print the results
print ("Status: ", LpStatus[prob.status])
Status:
Optimal
for v in prob.variables():
print (v.name, "=", v.varValue)
```

$x1 = 0.50856495$
$x2 = 0.39676692$
$x3 = 1.2650691$
$x4 = 0.29370832$
$y1 = 0.047353193$
$y2 = 1.1101106$

```
print
("The optimal value of  the objective function  is  =   ",
value(prob.objective))
```

The optimal value of the objective function is = 7.485201149079998
Solution for the given objective function for $q = 0$ and corresponding to above values of x_1, x_2, x_3 and x_4 is as under:

```
minz=((pow(2,1-q)*pow(3,q)*pow(x1,2)*pow(x2,-1)*pow(x3,1)*pow(x4,-1))+
(pow(4,1-q)*pow(4.2,q)*pow(x1,-3)*pow(x2,2)*pow(x3,-2)))
smpl = simplify(minz)
smpl


8.60711873476124
```

In table 1, optimal solutions are obtained corresponding to some values of q. The optimal solution can be obtained for other intermediate values of q ranging between 0 and 1 according to the requirement of the decision maker.

Table 1. Solutions of PGPP for different values of q

q	x_1	x_2	x_3	x_4	Solution of Linear System	Optimal Objective Value
0.0	0.5086	0.3968	1.2651	0.2937	7.4852	8.6071
0.1	0.5062	0.3971	1.2608	0.2929	7.6780	8.08592
0.2	0.5037	0.3970	1.2557	0.2921	7.8742	9.1181
0.3	0.5011	0.3968	1.2496	0.2912	8.0733	9.3863
0.4	0.4985	0.3962	1.2429	0.2902	8.2753	9.6644
0.5	0.4958	0.3956	1.2352	0.2892	8.4803	9.9505
0.6	0.4932	0.3947	1.2264	0.2882	8.6878	10.2457
0.7	0.4907	0.3936	1.2157	0.2872	8.8985	10.5508
0.8	0.4887	0.3922	1.2008	0.2862	9.1126	10.9693
0.9	0.4889	0.3892	1.1692	0.2854	9.3363	11.2111
1.0	0.4507	0.4199	1.3297	0.2626	10.0854	11.9235

5 Conclusion

The standard Posynomial Geometric programming problem with minimizing a posynomial objective function subject to posynomial inequality constraints were used in this research. Singh [13] proposed an alternative approach that involves transforming the posynomial constraints into exponential form to simplify the problem. In the case of a single objective Posynomial Geometric programming problem with interval coefficients, an interval-valued function could be used to represent uncertainty in the coefficients. An algorithm implemented here to solve the problem, and a Python code defines the objective function based on the interval coefficients. The algorithm can then be used to calculate the optimal solution for the posynomial objective function while considering the uncertainty in the coefficients. By transforming the posynomial constraints into exponential form using interval-valued functions, the approach provides a more efficient and accurate solution to the problem. Implementing the Python code to define the objective function based on interval coefficients allows for easy customization and adaptation to different scenarios.

References

[1] Beightler, C., & Phillips, D. (1976). Applied Geometric Programming. Wiley.

[2] Avriel, M., Dembo, R., & Passy, U. (1975). Solution of Generalized Geometric Programs. International Journal for Numerical Methods in Engineering , Vol. 9.

[3] Duffin, R., & Peterson, E. (1973). Geometric Programming with Signomials. Journal of Optimization Theory and Applications , 11 No.1.

[4] Liu, S.-T. (2006). Posynomial Geometric Programming with parametric uncertainty. European journal of operational research, 168, 345-353.

[5] Liu, S.-T. (2008). Posynomial Geometric Programming with interval exponents and coefficients. European journal of operational research, 186, 17-27.

[6] Ojha, A., & Biswal, K. (january 2010). posynomial geometric programming problems with multiple parameters. Journal of computing, vol. 2 (issue 1).

[7] Mahapatra, G., & Mandal, T. (2012). Posynomial parametric geometric programming with interval valued coefficient. J Optim Theory Appl , 154, 120-132.

[8] Ojha, A., & Biswal, K. (2010). Multi-objective Geometric Programming problem with weighted mean method. International Journal of Computer science and Information security, vol. 7 (no. 2).

[9] Mousavi, Z., & Saraj, M. (2019). Multi-objective Geometric Programming with interval coefficients: A Parametric approach. Earthline journal of Mathematical Sciences, vol. 2 (no. 2), 395-407.

[10] Das, P., & Roy, T. K. (2014). Solving a Multi-Objective Geometric Programming and its Application in Gravel box Problem. Journal of Global Research in Computer Science , 5 no. 7.

[11] Ojha, A., & Ota, R. R. (2015). A Hybrid Method for Solving Multi-objective Geometric Programming Problem. Int. J. Mathematics in Operational Research, Vol. 7 (No. 2).

[12] Oz, E., Guzel, N., & Alp, S. (2017). An Alternative Approach to the solution of Multi-Objective Geometric Programming Problem. Open Journal of Optimization, 6, 11-25.

[13] Singh, H., & Singh, A. (2023). Estimating Solution of Posynomial Geometric Programming Problems having Interval Coefficients. Journal for Educators, Teachers and Trainers , 14 (6).

[14] Abdullah, R. M., & Rasul, H. M. (2022). New Techniques of weighted sum method for solving multi-objective geometric programming problems. Journal of university of Babylon , vo. 30, No. 3, 168-181.

[15] Amuji, H. O., Nwachi, C. C., Okechukwu, B. N., Okeoma, I. O., & Inah, S. A. (2023). Approximating linear programming by geometric programming and its application to urban planning. African Journal of Mathematics and Statistics Studies , volume 6, issue 3, 115-127.

[16] Mondal, T., Ojha, A. K., & Pani, S. (2022). Solving Geometric Programming Problems with Triangular and Trapezoidal Uncertainty Distributions. RAIRO-Operrations Research , 56, 2833-2851.

Author information

Harpreet Singh, Research Scholar, Desh Bhagat University, Mandi Gobindgarh & Assistant Professor, Department of Mathematics, Guru Nanak College, Budhlada (Mansa), India.
E-mail: preethar_singh@yahoo.co.in

Amanpreet Singh, Assistant Professor, Department of Mathematics, GSSDGS Khalsa College, Patiala, India.
E-mail: drapsingh25@gmail.com