Nigerian Research Journal of Engineering and Environmental Sciences 2(2) 2017 pp. 588-595

Nigerian Research Journal of Engineering and Environmental Sciences
Journal homepage: www.rjees.com

Original Research Article

AN OBJECT ORIENTED APPROACH TO GEOMETRIC PROGRAMMING

***Ebojoh, E., Akpobi, J.A. and Tizhe, C.**

Department of Production Engineering, Faculty of Engineering, University of Benin, Benin City, Nigeria
*voke.ebojoh@uniben.edu

ARTICLE INFORMATION

Article history:
Received 19 December 2017
Revised 25 December, 2017
Accepted 26 December, 2017
Available online 29 December, 2017

Keywords:
Visual basic
Geometric programming
Resource allocation
Nonlinear problems
Mathematical modelling

ABSTRACT

Geometric programming is an efficient tool for solving nonlinear problems. This work developed and tested a Visual Basic programming language to solve the unconstrained geometric programming problems. The analysis of the result obtained from the Visual Basic programmed Language showed no significant error, whereas the manually calculated solution contained about 0.5% error. These results show that resource allocations on demand can be altered using the manual calculations and this will as well give a false figure on demand necessary for optimality. By the use of this program, the problems encountered in manual calculations can be eliminated.

1. INTRODUCTION

Profit maximization is an important issue for firms that pursue the most possible profit. Profit maximization problems traditionally is solved by differentiating with respect to input prices. Different methods or approaches from the traditional method has been applied through research, of which geometric programming (a mathematical programming technique) to derive the maximum value of profit with or without constrains was employed by Shiang-Tai et al (2004).

Tamilarasi et al. (2005) stated that geometric programming technique was derived from inequalities rather than calculus and it was given the name geometric programming because the geometric arithmetic mean inequality was the basis of its development. Geometric programming was developed in connection with problems of engineering design. The solution to engineering design problems requires a compromise among many alternatives which the engineer seeks an ideal compromise that optimizes some specific characteristic of design which depend on fixed and adjustable parameters. In

a typical optimal design, the system performs a given function at a minimum cost, and this is a stimulus for the development of geometric programming. Scott *et al.* (2004) described geometric programming as a well-established branch of optimization theory and very instrumental in the solution of many nonlinear problems occurring in diverse areas such as marketing, water pollution management, production engineering, transportation planning and machine maintenance. Shiang-Tai et al (2004) further explained that geometric programming was initially concerned with finite dimensional optimization problems where both the objectives and constrains were in polynomial form (i.e. polynomials with positive coefficient). Subsequently the theory was extended by Duffin et al (1967) to any finite dimensional convex programming problem which is termed geometric programming. As researches are intensified, geometric programming has been used for applications in mechanical and civil engineering, chemical engineering, probability and statistics, finance and economics, control theory, circuit design, information theory, coding and signal processing, wireless network, others are stochastic sensitivity analysis (Dupacova 2010), further Scott *et al.* (2004) says that GP can be applied in water pollution management, production engineering, transportation planning, machine maintenance and fields of sciences, also in project management (Ojha and Das, 2010).

This work makes an attempt to ease the stress associated with manual solving of geometric programming problems by developing a software using visual basic.net programming language to give more accurate optimal solution to the problem. This computer software will help in the following ways:

- Enhance easy computation by eliminating the stress encounter in manually solving of geometric programming problems.
- Save time lost used in manual solving of geometric programming problems.
- Reduce error generation using manual solving of geometric programming problems by providing accurate and efficient solution.

2. METHODOLOGY

The .net framework is an object oriented programming language, which makes use of objects which are a combination of data and codes. Data represent the relevant information (equations) that need to be programmed which in this case is the computation of our objective function (Z) and our basic variables (x_i). The codes denote the series of instructions which is written, manipulated and developed to get the data needed. This programming language is interpretive.

2.1. Mathematical Modeling/Algorithm

Normally, we have constrained and un-constrained geometric programming problems. The constrained geometric programming problem in general is defined as follows:

$$min.\, f(x) = \sum_{j=1}^{n} c_{oj} \prod_{i=1}^{m} x^{a0ij} \quad (1)$$

$$St.\ g_k(x) \equiv \sum_{j=1}^{nk} c_{kj} \prod_{i=1}^{m} x^{akij} \leq 1, \quad (2)$$

$$where\ x_i\ (i=1, 2\ \ldots m) > 0 \quad (3)$$

The constraints $C_{oi}(j = 1, 2. . .n_o)$ *and* $C_{kj}\ (k = 1, 2 . . ., n_k)$ are strictly positive, the exponents $a_{oij}(I = 1, 2, . . ., m, j = 1, 2, . . .,n_o)$ *and* $a_{kij}\ (k=1, 2, . . .,N, i=1, 2, . . .m, j=1, 2, . . .,n_k)$ are real numbers, where n indicates the total number of constraints, and n_o represents the number of terms in the objective

function, n_k denotes the number of terms in the k^{th} constraints (Tamilarasi et al., 2005). For the purpose of this work, unconstrained geometric programming is described.

2.2. Unconstrained Geometric Programming Algorithm

Taha (2002) stated that, geometric programming deals with the problems which the objective and the constraint functions are of the type:

$$Z = f(x) \sum_{j=1}^{N} U_j \quad (4)$$

$$Where\ u_j = C_j \prod_{i=1}^{n} u_j x^{aij},\ j=1, 2, \ldots, N \quad (5)$$

$C(j=1, 2, \ldots, n) > 0$, and $a_{ij}(i=1, 2, \ldots, n; j=1, 2, \ldots, N)$ are unrestricted in sign and C= coefficient; x = variables; a = exponents (unrestricted in sign).

The function $f(x)$ takes the form of a polynomial, but the exponents a_{ij} are unrestricted in sign, (i.e. they can be positive or negative) and all $C_j > 0$, $f(x)$ is called a posynomial.

Let's consider the minimization of the posynomial function $f(x)$ which is refer to as a primal function, the first partial derivative of Z must vanish at a point. Thus:

$$\frac{\partial z}{x_k} = \sum_{j=1}^{n} \frac{\partial u}{x_k} = \sum_{j=1}^{n} c_j a_{jk} (x_k)^{akj-1} \prod_{i \neq k} x_i^{a_{ik}} = 0 \quad (6)$$

$K = 1, 2, \ldots, n$ and because $x_k > 0$ by assumption

$$\frac{\partial z}{x_k} = 0, = \frac{1}{x_k} \sum_{j=1}^{N} a_{ki}\, u_j,\ k = 1, 2, \ldots, \quad (7)$$

Let Z^* = minimum value of Z, then $Z^* > 0$, because Z is a posynomial and each $x_k > 0$.
Define

$$y_j = \frac{u_j^*}{z^*} \text{ where } y_j > 0 \text{ and } \sum_j^n y_j = 1. \quad (8)$$

The value y_j represent the relative contribution of u_j to the optimal value of the objective function Z^*. The necessary conditions can now be written as:

$$\sum_{j=1}^{n} a_{kj}\, y_j = 0,\ k = 1, 2, \ldots, n \quad (9)$$

$$\sum_{j=1}^{n} y_j = 1,\ y_j > 0, \forall_j. \quad (10)$$

These are known as the orthogonality and normality conditions and will yield a unique solution for y_j, if n + 1 = N and all the equations are independent. Given the values of y_j^*, the values of Z^* and X^* can be determined as follows:

$$Z^* = (Z^*)^{\sum_{j=1}^{\infty} y_j}, \quad (11)$$

Because:

$$Z^* = \frac{U_j^*}{y_j^*}, \quad (12)$$

It follows that:

$$Z^* = \left(\frac{U_1^*}{y_1^*}\right)^{y_1^*} \left(\frac{U_2^*}{y_2^*}\right)^{y_2^*} \ldots\ldots\ldots \left(\frac{U_N^*}{y_N^*}\right)^{y_N^*} \quad (13)$$

$$Z^* = \left\{ \prod_{j=1}^{N} \left(\frac{C_j}{y_j^*}\right)^{y_j^*} \right\} \left\{ \prod_{j=1}^{N} \left(\prod_{i=1}^{n} (x_i)^{a_{ij}} \right)^{y_j^*} \right\} \quad (14)$$

$$Z^* = \left\{ \prod_{1}^{N} \left(\frac{C_j}{y_j^*}\right)^{y_j^*} \right\} \left\{ \prod_{i=1}^{n} (x_i^*)^{\sum_{j=1}^{n} a_{ij} y_j^*} \right\} \quad (15)$$

$$Z^* = \sum_{j=1}^{N} \left(\frac{C_j}{y_j^*}\right)^{y_j^*} \quad (16)$$

This step is justified because $\sum_{j=1}^{N} a_{ij}\, y_j^* = 0$, and anything to power zero is equals one. The value of Z^* is known once all y_j have been determined. Given y_j^* and Z^*, $U_j^* = y_j^* Z^*$ can be determined. Solution of the following equation then gives Equation 17.

$$U_j^* = C_j \prod_{i=1}^{n} (x_i^*)^{a_{ij}} \text{ j =1, 2 . . . N.,} \quad (17)$$

These are the basic equations on which the software is being developed.

2.3. The Program

Building a window based application with visual basic.net involves three basic steps:

- Creating the user interface
- Writing the program codes
- Running the program

2.3.1. Creating the user interface

This is also referred to as the design of the interface. The design is subjective, depending on the nature of the program and the programmer. This involves the use of forms, controls on the toolbox (e.g. labels, textbox, picture box, group box, buttons etc). All these put in place when needed make it to be user friendly and easily accessible. Also the properties must be set. These properties are programmable characteristics associated with forms and their controls. The settings of these properties are also subjective. These include the size, the font, background color and fore color etc (Deitel and Deitel, 2011).

2.3.2. Writing the program codes

The program codes are usually written in the code editor window. To write the codes, the basic language syntax rule associated with the .net framework was followed. Writing program codes gives more control on how the program will work (Deitel and Deitel, 2011). In writing the codes for the GP, the various formulae equation 1 to equation 17 considered herein were converted to visual basic .net programming language. The procedure for solving a GP problem can be explained through a flow chart as shown in Figure 1.

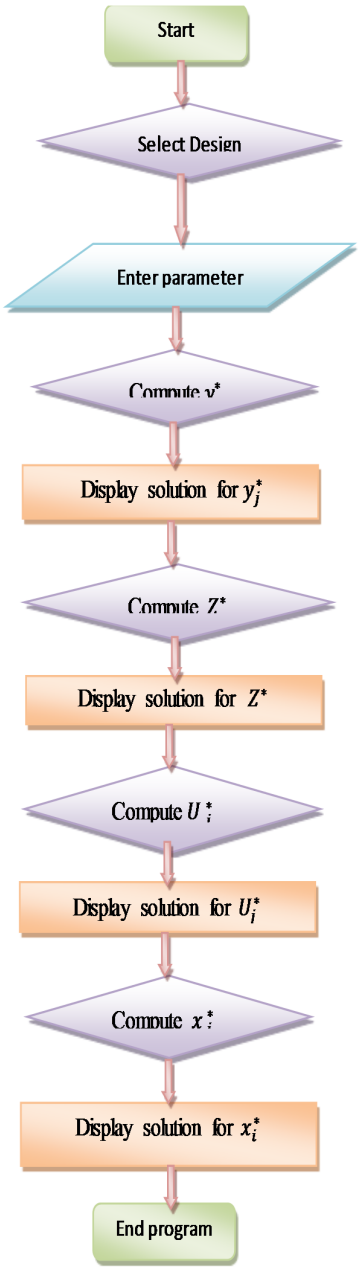

Figure 1: Flowchart of geometric programming software

In solving GP using the software, it is only in the run window or the interface that the user can operate the program and this can be achieved by clicking the debug button or the run icon in the menu bar or pressing F8 button on the keyboard. Logically then the software was tested on some practical problems involving GP problems based on the mathematical models used in designing the software. From the foregoing, the interface is user friendly and hence new users can easily use and manipulate the software to full capacity. Some features of the user friendliness can be illustrated from the

examples presented which are purely posynomial.

2.4. Numerical Examples

To illustrate the working of this software, the following examples were considered.

Example 1: $Min.Z = 7x_1x_2^{-1} + 3x_2x_3^{-2} + 5x_1^{-3}x_2x_3 + x_1x_2x_3,\ x_1x_2x_2 \geq 0$

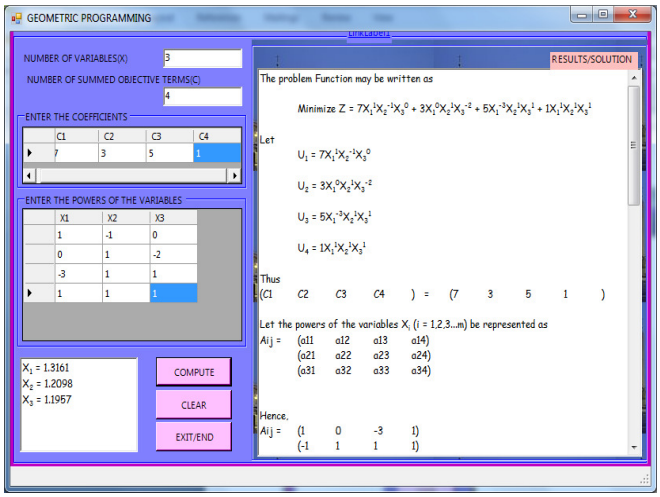

Figure 2: Output results for Example 1

Example 2: $Min.Z = 2x_1^2x_2^{-3} + 8x_1^{-3}x_2 + 3x_1x_2,\quad x_1x_2 \geq 0$

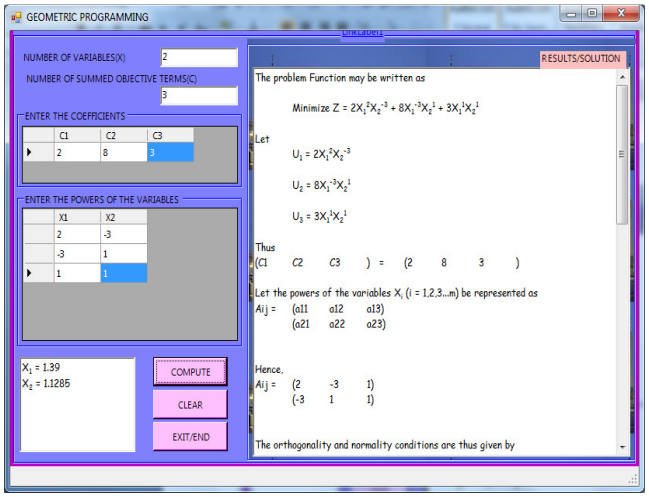

Figure 3: Output results for Example 2

Example 3: $Min.Z = 5x_1x_2^{-1}x_3^2 + x_1^{-2}x_3^{-1} + 10x_2^3 + 2x_1^{-1}x_2x_3^{-3}$

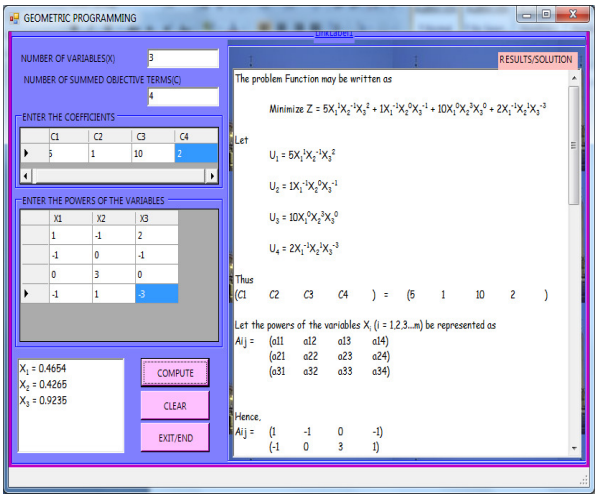

Figure 4: Output results for Example 3

3. RESULT AND DISCUSSION

The software is robust in that it gives a step by step manual solution which can be compared with bench mark questions and solutions. To the right of the software is the manual solution while to the left is a direct computation if one is not interested in a step wise solution but just the optimal results.

3.1 Result Comparism

The unique and optimal solutions that were calculated manually and the results computed by the programming language are compared in the Table 1.

Table 1: Comparism of result of manual and software results

Variable	Manual	Software	Difference	Error (%)
y_1^*	0.5000	0.5000	0.000	0.000
y_2^*	0.1667	0.1667	0.000	0.000
y_3^*	0.2083	0.2083	0.000	0.000
y_4^*	0.125	0.125	0.000	0.000
Z^*	15.310	15.230	0.080	0.523
U_1^*	7.655	7.615	0.040	0.525
U_2^*	2.550	2.5384	0.0116	0.457
U_3^*	3.190	3.173	0.017	0.536
U_4^*	1.910	1.9038	0.062	0.326
x_1^*	1.315	1.3161	0.011	0.084
x_2^*	1.202	1.2098	0.078	0.645
x_3^*	1.210	1.1957	0.0143	1.196

4. CONCLUSION

The percentage error on the unique values of y_j is zero, while for the unique values of U_j is at the average of 0.461. The average error percentage for the optimal solution of x_i is 0.642. This is due to the difference between the value of Z* obtained that has a percentage of 0.523, thus it can be observed that the computed result from the program is more accurate than the manual calculation. This program has limited error from the result analysis in were the manually calculated solution contain about 0.5% error. From the results, it shows that resource allocations on demand can be altered using the manual calculation and this will as well give a false demand necessary for optimality. By the use of this program, all the problems met in manually calculated results could be eliminated.

5. CONFLICT OF INTEREST

There is no conflict of interest associated with this work.

REFERENCES

Deitel, P.J. and Deitel, H.M. (2011). Visual basic 2010: How to program. Pearson Education Inc. USA. p. 93.

Duffin, R.J., Peterson, E.L., and Zener, C. (1967). *Geometric Programming: Theory and Application.* John Wiley and Sons Inc.

Dupacova, J. (2010). Stochastic geometric programming with an application. *Kybernetika*, 46(3), pp. 374-386.

Ojha, A.K. and Das, A.K. (2010). Geometric programming problems with coefficients and exponents associated with binary numbers. *International journal of computer science,* 7(1) pp. 49-55

Scott, C.H., Jefferson, T.R. and Jorjani, S. (2004). Duals for classical inventory models via geometric generalized programming. *Journal of applied mathematics and decision science,* 8(3), pp 191-200

Shiang-Tai L. and Chiang K. (2004), Solving fuzzy transportation problems based on extension principle. *European Journal of Operations Research,* 153, pp. 661-674.

Taha, H.A. (2002). Operations Research: An Introduction. International Edition. 7th Ed. Pearson Education, India

Tamilarasi, A., Natarajan, A.M. and Balasubramani, P. (2005). Operations Research. Pearson Education, India. pp 731-723.