COMPUTER PROGRAMMING LANGUAGE SOLUTION TO A TRANSPORTATION PROBLEM INVOLVING A CONCAVE COST FUNCTION

Abstract

The work is on computer programming language solution to a transportation problem involving a concave cost function. Two computer programming languages; Wolfram Mathematica and Anaconda Python programming ~~softwares~~ tools were employed in this study to effectively solve four real life examples from published works. The results from the programming languages yielded an optimal value of ~~₦~~253,000 with an optimal solution as $z_{12} = 13,\ z_{22} = 5,\ z_{23} = 8,$ $z_{31} = 11\ and\ z_{33} = 4$ in the first example and the remaining three examples were successfully solved with optimal values of ~~₦~~377,000, GH¢ 236,000 and ~~₦~~509,000 respectively, and the results agreed with the results of existing Karush-Kuhn-Tucker (KKT) procedure of MODI method.

Keywords: Wolfram Mathematica, Anaconda Python, Concave Cost Function, Transportation Problem, Optimal Solution

1 Introduction

Transportation problems involving a concave cost function, simply means a nonlinear transportation problem; indicating a scenario whereby volume discountsare being available for bulk shipments. In this case, the cost function of the transportation problem is separable, and the marginal cost (cost per unit of goods shipped) decreases as the shipment volume increases, so it generally assumes a concave structure. Itincreasesdue to the increase in the total cost per additional unit of goods shipped (Haruna et al, 2012). Discounts may be directly related to the unit of commodity or may have the same rate for a particular amount. Thus, the discount may be either directly associated to the unit commodity or have the equivalent rate for some quantity. However, if the discount is directly associated to the unit commodity, then the resulting cost function becomes continuous and possesses continuous first partial derivatives.

Profit and cost is what virtually everyone deems interested in employing, using any efficient resources to optimize; hence various forms of transportation models are in existence. There are different kinds of transportation problems which are applied in the business world and the primary aim of a transportation problem is to find a means of moving this transfer of goods at a minimized total cost (Mostafa et al, 2022; Kaur & Kumar, 2011). In describing the transportation problem in its conventional form, the assumption is that an informed decision maker has an understanding on the value of transportation cost, demand and supply; hence, unpredictability is a common occurrence in real life circumstances.

Formatted: No Spacing, Left, Line spacing: single, Tab stops: Not at 3.25" + 5.55"

Comment [PR1]: Better to not use acronyms in the abstract.

Comment [PR2]: These are not germane to the problem being studied. They are just tools for implementing mathematics. If Mathcad and C++ were used, the results would be no different.

Have a look at the keywords of cited papers for ideas. Use the number of keywords allowed.

Comment [PR3]: Very confusing. Consider the illustration of "concave up" in this discussion: https://www.statisticshowto.com/inflection-point. I assume the author means "concave up" and not "concave down". If x is the volume shipped and y is the cost per unit of volume then "concave up" means cost will ultimately begin to rise once again. Thus "concave up" does not tell the whole story.

Comment [PR4]: Is this taking a stab at the rising side of "concave up"? Still very confusing. Please rethink this explanation since it is fundamental to the paper's discussion.

Perhaps an illustration would be useful. Show the curve and then explain its various components.

There are other affects on shipping cost. One is distance. Another is cargo fragility. Still another could be single-unit size (a large wide farm tractor, for instance).

Comment [PR5]: This is true from a shipper's perspective. From a hauler's perspective, I suspect the primary aim is profit. Cost vs. price. At balance, there win-win potential between shipper and hauler.

This study ~~is channeled to~~ tackles a transportation problem in a concave cost function usingcomputer programming languages. However, there are some factors that are responsible for the cost of goods, and some of them are transport, raw materials' costs and labour. This implies that the cost of raw materials is directly proportional to the cost of the goods, and the pricing system is also affected when there is a significant variation in the transportation cost (Rudi et al, 2016). The cost of goods per unit shipped is assumed to be constant irrespective of the quantity shipped from a given source to a defined destination; but the cost sometimes may not be constant in actuality. Sometimes, quantity discounts are feasible for large shipments in such a way that the marginal cost of transporting a unit might approach a specific pattern (Minken & Johansen, 2019).

2 Review of Related Literature

Oliveira et al. (2020) carried out a research on a comparative study of linear programming and nonlinear programming models of the ship speed optimisation problem in maritime transportation. It was demonstrated in the work reported in the study that the linear programming formulation yielded better results than the nonlinear programming formulation for the ship speed optimisation problem. Therefore, great care must be taken in using the uniform single speed assumption, in view of the complex nature of the functional relation between ship fuel consumption rate and ship speed, as pointed out by Psarftis& Kontovas (2013). In view of the findings summarised in the work, the linear programming formulation of the ship speed optimisation problem holds many promises that worth~~y of~~ exploring in further research. These include complex functional relations between ship fuel consumption and ship speed to take into account dependence on ship payload and weather conditions encountered in ship voyages, ship routing in both liner and tramp shipping, and ship fuel bunkering.

Amaravathy et al. (2018) compared the existing methods with the one they worked on the optimum solution of OFSTF and MDMAmethods. In their study, different approaches to the OFSTF method (origin, first, second, third, fourth quadrants) were used to directly obtain a practical solution to the transport problem. The proposed method was unique and always provided a practical (probably optimal) solution without disturbing the state of degeneration. This process involved a minimum of iterations to achieve the optimization. In order to confirm the validity of the proposed method, they solved numerical examples and discussed the degeneracy problem.

3 Transportation Problem via Concave Cost Functions

Volume discounts may be available for bulk shipments. In this case, the cost function of the transportation problem is separable, and the marginal cost (cost per unit of goods shipped) decreases as the shipment volume increases, so ~~it is~~shipping cost generally assumes a concave structure. Itincreasesdue to the increase in the total cost per additional unit of goods

Comment [PR6]: "However" ? What is said next is true whether or not programming languages are involved. It would be better to say that a computer program was written to consider the following factors ……..

Should also mention the analysis methods used. Give full names and a citation on the basic methods employed. Should also briefly state some background on the methods.

Comment [PR7]: Do you mean "speed of transport" ?

Comment [PR8]: Although you do not mean "linear", I keep reading that instead of "liner". It will be the same for others who read this paper. Can a different word be used here? Perhaps "corporate" ?

Comment [PR9]: Always spell-out acronyms on first use. Should also provide a citation on their basic use. Thinking in this way broadens the applicability of the work beyond one's immediate peer group. Thus the work rises in value.

Comment [PR10]: Cost per unit shipped vs. Cost per volume shipped is pretty well confused at this point. More clarity is needed.

shipped(Haruna et al, 2012). However, If the discount is directly associated to the unit commodity, then the resulting cost function becomes continues and possesses continues first partial derivatives.

4 Computer Written Programming Codes for Transportation Problem in a Concave Cost Function

Comment [PR11]: The problem to be solved and its parameters should be explained first. Then the method of analysis. Then the associated mathematics, equations given in the sequence applied.

Equation parameters should be explained in terms of the problem to be solved. Results should be given in terms of the target, ex: profit, cost, ….. What parameter values result in the best result?

Programming implementation of equations and their runtime outputs should go in an appendix. But, they are very instructive.

This section explains how the transportation problem in a concave cost function could be solved using two programming languages codes written in Wolfram Mathematica programming and Anaconda Python programming.

4.1 Wolfram Mathematica Computer Written Programming Codes for Transportation Problem in a Concave Cost Function

```
Clear[f2, z, k, R, C, L, K, V, a];
m=p (Number of sources);
n=q (Number of destinations);
f2[z_ ]:= k11*z[1,1] – p11*z[1,1]^2+k12*z[1,2] – p12*z[1,2]^2+…+k1j*z[1,j] – p1j*z[1,j]^2+…+
        k1n*z[1,n] – p1n*z[1,n]^2+k21*z[2,1] – p21*z[2,1]^2+k22*z[2,2] – p22*z[2,2]^2+ k2j*z[2,j]
        – p2j*z[2,j]^2…+k2n*z[2,n] – p2n*z[2,n]^2+…+ki1*z[i,1] – pi1*z[i,1]^2+ki2*z[i,2] –
        pi2*z[i,2]^2+…+kij*z[i,j] – pij*z[i,j]^2+…+kin*z[i,n] – pin*z[i,n]^2+…+km1*z[m,1] –
        pm1*z[m,1]^2+km2*z[m,2] – pm2*z[m,2]^2+…kmj*z[m,j] – pmj*z[m,j]^2+…+kmn*z[m,n]
        – pmn*z[m,n]^2;
R1:=z[1,1]+z[1,2]+…+z[1,j]+…+z[1,n]==S1;
R2:=z[2,1]+z[2,2]+…+z[2,j]+…+z[2,n]==S2;
⋮          ⋮          ⋮     ⋮
Ri:=z[i,1]+z[i,2]+…+z[i,j]+…+z[i,n]==Si;
⋮          ⋮          ⋮     ⋮
Rm:=z[m,1]+z[m,2]+…+z[m,j]+…+z[m,n]==Sm;
C1:=z[1,1]+z[2,1]+…+z[i,1]+…+z[m,1]==D1;
C2:=z[1,2]+z[2,2]+…+z[i,2]+…+z[m,2]==D2;
⋮          ⋮          ⋮     ⋮
Cj:=z[1,j]+z[2,j]+…+z[i,j]+…+z[m,j]==Dj;
⋮          ⋮          ⋮     ⋮
Cn:=z[1,n]+z[2,n]+…+z[i,n]+…+z[m,n]==Dn;
L=Join[{R1},{R2},…,{Ri},…,{Rm},{C1},{C2},…,{Cj},…,{Cn}];
K1:=z[1,1]>=0;
K2:=z[1,2]>=0;
⋮     ⋮
K3:=z[1,j]>=0;
⋮     ⋮
K4:=z[1,n]>=0;
K5:=z[2,1]>=0;
K6:=z[2,2]>=0;
⋮     ⋮
```

K7:=z[2,j]>=0;
⋮ ⋮
K8:=z[2,n]>=0;
⋮ ⋮
K9:=z[i,1]>=0;
K10:=z[i,2]>=0;
⋮ ⋮
K11:=z[i,j]>=0;
⋮ ⋮
K12:=z[i,n]>=0;
⋮ ⋮
K13:=z[m,1]>=0;
K14:=z[m,2]>=0;
⋮ ⋮
K15:=z[m,j]>=0;
⋮ ⋮
K16:=z[m,n]>=0;
K=Join[{K1},{K2},…,{K3},…,{K4},{K5},{K6},…,{K7},…,{K8},…,{K9},{K10},…,{K11},
…,{K12},…,{K13},{K14},…,{K15},…,{K16}];
V=Join[L,K];
a=Variables[f2[z]];
`tim=Timing[NMinimize[{f2[z],V},a,StepMonitor`:→Print["Step:

z[1,1],z[1,2],z[1,3],z[3,3] = ",z[1,1]",",z[1,2]",",z[1,3]",",z[3,3]]]];

tim

4.2 Anaconda Python Computer Written Programming Codes for Transportation Problem in a Concave Case

In [1]:
```
from gekko import GEKKO
import numpy as np
from matplotlib import pyplot as plt
```

In [2]:
```
# Initialize Model
m = GEKKO(remote =True)
```

In [3]:
```
# help (m)
# define parameters
eq_1 = m.Param(value =S1)
eq_2 = m.Param(value =S2)
⋮ ⋮
eq_i = m.Param(value =Si)
⋮ ⋮
eq_m = m.Param(value =Sm)
eq_m+1 = m.Param(value =D1)
```

Comment [PR12]: Python is a programming language. Anaconda is a means of employing that language. There should be no essential difference in results whether Mathematica or Python is used to implement the equations and calculation processes being applied.

```
    eq_m+2 = m.Param(value =D2)
    ⋮          ⋮
    eq_j = m.Param(value =Dj)
    ⋮          ⋮
    eq_n = m.Param(value =Dn)
```

In [4]:
```
    # Initialize variables, k=number of variables
    z11,…,z1j,…,z1n,z21,…,z2j,z2n,…,zi1,zi2,…,zij,…,zin,…,zm1,zm2,…,\
zmj,…,zmn= [m.Var() for i in range(k)]
```

In [5]:
```
# Initialize values(Assigning values or constants whose summation is less than or equal
to Si or Dj)
    z11 = m.Var(value = c1)
    ⋮          ⋮
    z1j = m.Var(value = c2)
    ⋮          ⋮
    z1n = m.Var(value = c3)
    z21 = m.Var(value = c4)
    ⋮          ⋮
    z2j = m.Var(value = c5)
    z2n = m.Var(value = c6)
    ⋮          ⋮
    zi1 = m.Var(value = c7)
    zi2 = m.Var(value = c8)
    ⋮          ⋮
    zij = m.Var(value = c9)
    ⋮          ⋮
    zin = m.Var(value = c10)
    ⋮          ⋮
    Zm1 = m.Var(value = c11)
    Zm2 = m.Var(value = c12)
    ⋮          ⋮
    Zmj = m.Var(value = c13)
    ⋮          ⋮
    Zmn = m.Var(value = c14)
```

In [6]:
```
    z11.lower = 0
    ⋮          ⋮
    z1j.lower = 0
    ⋮          ⋮
    z1n.lower = 0
    z21.lower = 0
    ⋮          ⋮
    z2j.lower = 0
    z2n.lower = 0
    ⋮          ⋮
```

```
zi1.lower = 0
zi2.lower = 0
⋮          ⋮
zij.lower = 0
⋮          ⋮
zin.lower = 0
⋮          ⋮
Zm1.lower = 0
Zm2.lower = 0
zmj.lower = 0
⋮          ⋮
Zmn.lower = 0
```

In [7]:
```
z11.upper =None
⋮          ⋮
z1j.upper =None
⋮          ⋮
z1n.upper =None
z21.upper =None
⋮          ⋮
z2j.upper =None
z2n.upper =None
⋮          ⋮
zi1.upper =None
zi2.upper =None
⋮          ⋮
zij.upper =None
⋮          ⋮
zin.upper =None
⋮          ⋮
Zm1.upper =None
Zm2.upper =None
zmj.upper =None
⋮          ⋮
Zmn.upper =None
```

In [8]:
```
m.Equations([z11+z12+…+z1j+…+z1n == eq_1,…, zi1+zi2+zij+…+zin == eq_i,\
…,zm1+zm2+…+zmj+…+zmn == eq_m,z11+z21+…+zi1+…+zm1 == eq_m+1 \
…,z1j+z2j+…+zij+…+zmj == eq_j,…, z1n+z2n+…+zin+…+zmn == eq_n])
```

In [9]:
```
m.Obj(k11*z11 – p11*z11**2+ k12*z12 – p12*z12**2+…+ k1j*z1j – p1j*z1j**2+…+ \
k1n*z1n – p1n*z1n**2+ k21*z21 – p21*z21**2+ k22*z22 – p22*z22**2+ k2j*z2j –\
p2j*z2j**2…+ k2n*z2n – p2n*z2n**2+…+ ki1*zi1 – pi1*zi1**2+ ki2*zi2 –\
pi2*zi2**2+…+ kij*zij – pij*zij**2+…+ kin*zin – pin*zin**2+…+ km1*zm1 –\
pm1*zm1**2+ km2*zm2 – pm2*zm2**2+… kmj*zmj – pmj*zmj**2+…+ kmn*zmn –\
pmn*zmn**2)
```

```
In [10]: m.options.IMODE =3

In [11]: m.solve(GEKKO(remote=True))

In [12]: m.options.SOLVER=1

In [12]: m.options.SOLVER=1

In [13]: print('')
         print('Results')
         print('z11: ' + str(z11.value))
                    ⋮          ⋮
         print('z1j: ' + str(z1j.value))
                    ⋮          ⋮
         print('z1n: ' + str(z1n.value))
         print('z21: ' + str(z21.value))
                    ⋮          ⋮
         print('z2j: ' + str(z2j.value))
         print('z2n: ' + str(z2n.value))
                    ⋮          ⋮
         print('zi1: ' + str(zi1.value))
         print('zi2: ' + str(zi2.value))
                    ⋮          ⋮
         print('zij: ' + str(zij.value))
                    ⋮          ⋮
         print('zin: ' + str(zin.value))
                    ⋮          ⋮
         print('zm1: ' + str(zm1.value))
         print('zm2: ' + str(zm2.value))
         print('zmj: ' + str(zmj.value))
                    ⋮          ⋮
         print('zmn: ' + str(zmn.value))

         Results
```

5 Numerical Problems

Example 1

Unilever Nigeria Plc located in Apapa Ikeja, produces and sells the products as indicated in the Table 1 together with the associated company's percentage discount:

Table 1: Unit Cost of Products and Company's Percentage Discount

	MARKETS SEGMENTS			SUPPLY
	P	Q	R	
Omo washing powder	5	4	6	13,000
Blue Band margarine	7	6	5	13,000
Vaseline	9	11	8	15,000
DEMAND	11,000	18,000	12,000	
Company's percentage discount				
	P	Q	R	
Omo washing powder	0.03	0.015	0.04	
Blue Band margarine	0.02	0.03	0.05	
Vaseline	0.035	0.05	0.03	

Source:Okenwe (2018)

Due to the discount given to each box as a result of large volume of transporting from source i to destination j, the formulation of the transportation problem in nonlinear form is:

$$Min. \sum_{i=1}^{3}\sum_{j=1}^{3} k_{ij} z_{ij}$$

$$S.t. \quad z_{11} + z_{12} + z_{13} = 13$$
$$z_{21} + z_{22} + z_{23} = 13$$
$$z_{31} + z_{32} + z_{33} = 15$$
$$z_{11} + z_{21} + z_{31} = 11$$
$$z_{12} + z_{22} + z_{32} = 18$$
$$z_{13} + z_{23} + z_{33} = 12$$

Where

$k_{11}z_{11} = 5z_{11} - p_{11}z_{11}^2$, $k_{12}z_{12} = 4z_{12} - p_{12}z_{12}^2$, $k_{13}z_{13} = 6z_{13} - p_{13}z_{13}^2$, $k_{21}z_{21} = 7z_{21} - p_{21}z_{21}^2$

$k_{22}z_{22} = 6z_{22} - p_{22}z_{22}^2$, $k_{23}z_{23} = 5z_{23} - p_{23}z_{23}^2$, $k_{31}z_{31} = 9z_{31} - p_{31}z_{31}^2$, $k_{32}z_{32} = 11z_{32} - p_{32}z_{32}^2$

$k_{33}z_{33} = 8z_{33} - p_{33}z_{33}^2$

Comment [PR13]: Where do the constants come from?

Due to the discount given to each box as a result of large volume of transporting from source i to destination j, then the cost function (k_{ij}) is indicated as:

Comment [PR14]: I see the discount in the table but the table does not refer to volume purchased/shipped.

$$k_{11}z_{11} = 5z_{11} - 0.03z_{11}^2,\ k_{12}z_{12} = 4z_{12} - 0.015z_{12}^2,\ k_{13}z_{13} = 6z_{13} - 0.04z_{13}^2,\ k_{21}z_{21} = 7z_{21} - 0.02z_{21}^2$$
$$k_{22}z_{22} = 6z_{22} - 0.03z_{22}^2,\ k_{23}z_{23} = 5z_{23} - 0.05z_{23}^2,\ k_{31}z_{31} = 9z_{31} - 0.035z_{31}^2,\ k_{32}z_{32} = 11z_{32} - 0.05z_{32}^2$$
$$k_{33}z_{33} = 8z_{33} - 0.03z_{33}^2$$

The output of Example 1 via Wolfram Mathematica and Anaconda Python is shown below;

5.1 Wolfram Mathematica Output of Example 1

```
Clear[f2,z,L,K,V,a];
f2[z_]:=5*z[1,1]-0.03*z[1,1]^2+4*z[1,2]-0.015*z[1,2]^2+6*z[1,3]-0.04*z[1,3]^2+7*z[2,1]-
0.02*z[2,1]^2+6*z[2,2]-0.03*z[2,2]^2+5*z[2,3]-0.05*z[2,3]^2+9*z[3,1]-0.035*z[3,1]^2+11*z[3,2]-
0.05*z[3,2]^2+8*z[3,3]-0.03*z[3,3]^2;
L1:=z[1,1]+z[1,2]+z[1,3]==13;
L2:=z[2,1]+z[2,2]+z[2,3]==13;
L3:=z[3,1]+z[3,2]+z[3,3]==15;
L4:=z[1,1]+z[2,1]+z[3,1]==11;
L5:=z[1,2]+z[2,2]+z[3,2]==18;
L6:=z[1,3]+z[2,3]+z[3,3]==12;
L=Join[{L1},{L2},{L3},{L4},{L5},{L6}];
K1:=z[1,1]>=0;
K2:=z[1,2]>=0;
K3:=z[1,3]>=0;
K4:=z[2,1]>=0;
K5:=z[2,2]>=0;
K6:=z[2,3]>=0;
K7:=z[3,1]>=0;
K8:=z[3,2]>=0;
K9:=z[3,3]>=0;
K=Join[{K1},{K2},{K3},{K4},{K5},{K6},{K7},{K8},{K9}];
V=Join[L,K];
a=Variables[f2[z]];
tim=Timing[NMinimize[{f2[z],V},a,StepMonitor :→Print["Step: z[1,1],z[1,2],z[1,3],z[3,3] =
",z[1,1]",",z[1,2]",",z[1,3]",",z[3,3]]]];
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.679135 , 12.2714 , 0.0494333 , 1.27443
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.679135 , 12.2714 , 0.0494333 , 1.27443
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.00452482 , 12.7951 , 0.200366 , 0.533278
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.00452482 , 12.7951 , 0.200366 , 0.533278
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.00452482 , 12.7951 , 0.200366 , 0.533278
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.00452482 , 12.7951 , 0.200366 , 0.533278
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.00452482 , 12.7951 , 0.200366 , 0.533278
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.00452482 , 12.7951 , 0.200366 , 0.533278
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0761768 , 12.5756 , 0.348219 , 0.131187
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0610422 , 12.731 , 0.207952 , 0.0591868
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0610422 , 12.731 , 0.207952 , 0.0591868
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0610422 , 12.731 , 0.207952 , 0.0591868
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0610422 , 12.731 , 0.207952 , 0.0591868
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0464471 , 12.9467 , 0.00687596 , 0.365764
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0464471 , 12.9467 , 0.00687596 , 0.365764
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0920152 , 12.7977 , 0.110271 , 0.0139504
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0920152 , 12.7977 , 0.110271 , 0.0139504
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0546995 , 12.9373 , 0.00802395 , 0.0512077
```

```
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0812792 , 12.8856 , 0.0331009 , -0.00048798
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0812792 , 12.8856 , 0.0331009 , -0.00048798
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0361427 , 12.9633 , 0.000553165 , 0.0224059
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0361427 , 12.9633 , 0.000553165 , 0.0224059
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0520463 , 12.9405 , 0.00746324 , 0.00716501
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0614247 , 12.9315 , 0.0070737 , 0.00427768
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0614247 , 12.9315 , 0.0070737 , 0.00427768
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0614247 , 12.9315 , 0.0070737 , 0.00427768
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0505677 , 12.9469 , 0.00255937, 0.000755734
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.09301 , 12.8942 , 0.0128178 , 0.00827994
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0827176 , 12.9154 , 0.00191623, 0.000232425
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0827176 , 12.9154 , 0.00191623, 0.000232425
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.121709 , 12.8703 , 0.00795573, -0.00110205
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.196778 , 12.7822 , 0.0210317 , 0.00454851
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.196778 , 12.7822 , 0.0210317 , 0.00454851
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.207363 , 12.7865 , 0.00613269, -0.000462094
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.334976 , 12.6352 , 0.0298175 , -0.000108529
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.334976 , 12.6352 , 0.0298175 , -0.000108529
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.510508 , 12.4472 , 0.0422997 , 0.120491
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.488911 , 12.4803 , 0.0307975 , 0.0812445
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.488911 , 12.4803 , 0.0307975 , 0.0812445
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.488911 , 12.4803 , 0.0307975 , 0.0812445
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.488911 , 12.4803 , 0.0307975 , 0.0812445
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.462777 , 12.5068 , 0.0303988 , 0.0465901
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.462777 , 12.5068 , 0.0303988 , 0.0465901
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.462777 , 12.5068 , 0.0303988 , 0.0465901
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.462777 , 12.5068 , 0.0303988 , 0.0465901
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.451278 , 12.5173 , 0.0314083 , -0.00130412
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.451278 , 12.5173 , 0.0314083 , -0.00130412
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454816 , 12.52 , 0.0251821 , 0.0191165
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454816 , 12.52 , 0.0251821 , 0.0191165
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454816 , 12.52 , 0.0251821 , 0.0191165
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454816 , 12.52 , 0.0251821 , 0.0191165
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454816 , 12.52 , 0.0251821 , 0.0191165
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454816 , 12.52 , 0.0251821 , 0.0191165
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454816 , 12.52 , 0.0251821 , 0.0191165
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454816 , 12.52 , 0.0251821 , 0.0191165
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454419 , 12.5258 , 0.0197449 , 0.0185168
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454419 , 12.5258 , 0.0197449 , 0.0185168
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.454419 , 12.5258 , 0.0197449 , 0.0185168
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.445135 , 12.5355 , 0.0193661 , 0.00427392
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.441966 , 12.541 , 0.0170058 , 0.00261565
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.441447 , 12.5459 , 0.0126755 , 0.00311075
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.441447 , 12.5459 , 0.0126755 , 0.00311075
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.441447 , 12.5459 , 0.0126755 , 0.00311075
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.441447 , 12.5459 , 0.0126755 , 0.00311075
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.441447 , 12.5459 , 0.0126755 , 0.00311075
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.437815 , 12.5501 , 0.012064 , 0.0000337962
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.437815 , 12.5501 , 0.012064 , 0.0000337962
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.441787 , 12.5518 , 0.00639919 , 0.00902117
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.436281 , 12.5617 , 0.00201731 , 0.00303067
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.436281 , 12.5617 , 0.00201731 , 0.00303067
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.436281 , 12.5617 , 0.00201731 , 0.00303067
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.442416 , 12.5576 , 0. 0.0111263
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.437073 , 12.5629 , 0. 0.0029804
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.437073 , 12.5629 , 0. 0.0029804
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.437073 , 12.5629 , 0. 0.0029804
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.438726 , 12.5613 , 0. 0.001833
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.438726 , 12.5613 , 0. 0.001833
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.438726 , 12.5613 , 0. 0.001833
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.438726 , 12.5613 , 0. 0.001833
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.439106 , 12.5609 , 0. 0.000126832
```

```
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.446625 , 12.5534 , 0. 0.00368506
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.446625 , 12.5534 , 0. 0.00368506
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  -1.08549 , 14.0855 , 0. 0.00451078
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.131384 , 12.8686 , 0. -0.00737012
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0027845 , 12.9972 , 0. -0.0015764
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  3.05114*10^-6 , 13. , 0. -0.0000134125
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  2.379*10^-12 , 13. , 0. -1.56092*10^-11
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  1.19982*10^-23 , 13. , 0. 0.
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.446625 , 12.5524 , 0.001 , 0.00368506
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.588814 , 12.4074 , 0.00376971 , 0.995278
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.00588814 , 12.8423 , 0.151809 , 1.52872
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0251432 , 12.9023 , 0.0725069 , 1.84993
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0211205 , 12.9227 , 0.0562188 , 3.17547
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.0121676 , 12.9687 , 0.0191761 , 4.08295
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0.000229113 , 12.9993 , 0.000433091 , 4.00134
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  2.30458*10^-6 , 13. , 4.41297*10^-6, 4.00001
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 3.69684
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 3.92754
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 3.98208
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 3.99553
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 3.99888
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 3.99972
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 3.99993
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 3.99998
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 4.
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 4.
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 4.
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 4.
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 4.
Step : z[1,1],z[1,2],z[1,3],z[3,3] =  0. 13. , 0. 4.
tim
{6.09964,{241.8,{z[1,1]->0.,z[1,2]->13.,z[1,3]->0.,z[2,1]->0.,z[2,2]->5.,z[2,3]->8.,z[3,1]-
>11.,z[3,2]->9.24835*10^-8,z[3,3]->4.}}}
Clear[f,fmin]
f2[z]=5*z[1,1]+4*z[1,2]+6*z[1,3]+7*z[2,1]+6*z[2,2]+5*z[2,3]+9*z[3,1]+11*z[3,2]+8*z[3,3];
z[1,1]=0;
z[1,2]=13;
z[1,3]=0;
z[2,1]=0;
z[2,2]=5;
z[2,3]=8;
z[3,1]=11;
z[3,2]=0;
z[3,3]=4;
fmin=f2[z];
fmin
253
```

5.2 Anaconda Python Output of Example 1

```
In [1]:  from gekko import GEKKO
         import numpy as np
         from matplotlib import pyplot as plt
In [2]:  # Initialize Model
         m = GEKKO(remote =True)
In [3]:  # help (m)
         # define parameters
         eq_1 = m.Param(value =13)
         eq_2 = m.Param(value =15)
         eq_3 = m.Param(value =11)
         eq_4 = m.Param(value =18)
         eq_5 = m.Param(value =12)
In [4]:  # Initialize variables
         z11, z12, z13, z21, z22, z23, z31, z32, z33 = \
         [m.Var() for i in range(9)]
In [5]:  # Initialize values
         z11 = m.Var(value = 1)
         z12 = m.Var(value = 2)
```

```
          z13 = m.Var(value = 3)
          z21 = m.Var(value = 2)
          z22 = m.Var(value = 2)
          z23 = m.Var(value = 5)
          z31 = m.Var(value = 1)
          z32 = m.Var(value = 5)
          z33 = m.Var(value = 4)
In [6]:   z11.lower = 0
          z12.lower = 0
          z13.lower = 0
          z21.lower = 0
          z22.lower = 0
          z23.lower = 0
          z31.lower = 0
          z32.lower = 0
          z33.lower = 0
In [7]:   z11.upper =None
          z12.upper =None
          z13.upper =None
          z21.upper =None
          z22.upper =None
          z23.upper =None
          z31.upper =None
          z32.upper =None
          z33.upper =None
In [8]: m.Equations([z11+z12+z13 == eq_1, z21+z22+z23 == eq_1, z31+z32+z33 == eq_2,\ z11+z21+z31
        == eq_3, z12+z22+z32 == eq_4, z13+z23+z33 == eq_5])
Out [8]:  [<gekko.gekko.EquationObj at 0x237ff8d28e0>,
          <gekko.gekko.EquationObj at 0x237ff8d2c10>,
          <gekko.gekko.EquationObj at 0x237ff8d29d0>,
          <gekko.gekko.EquationObj at 0x237ff8cbbb0>,
          <gekko.gekko.EquationObj at 0x237814de4f0>,
          <gekko.gekko.EquationObj at 0x237ff8db670>]
In [9]:   m.Obj(5*z11-0.03*z11**2+4*z12-0.015*z12**2+4*z13-0.04*z13**2+7*z21-0.02*z21**2\
          +6*z22-0.03*z22**2+5*z23-0.05*z23**2+9*z31-0.035*z31**2+11*z32-0.05*z32**2\
          +8*z33-0.03*z33**2)
In [10]:  m.options.IMODE =3

In [11]:  m.solve(GEKKO(remote=True))
apm 105.112.210.32_gk_model0 <br><pre> ---------------------------------------------------------
------
 APMonitor, Version 1.0.1
 APMonitor Optimization Suite
 ----------------------------------------------------------------

 --------- APM Model Size ------------
 Each time step contains
   Objects      :              0
   Constants    :              0
   Variables    :             32
   Intermediates:              0
   Connections  :              0
   Equations    :              7
   Residuals    :              7

 Number of state variables:             27
 Number of total equations: -            6
 Number of slack variables: -            0
 ----------------------------------------
 Degrees of freedom       :             21

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  1.6833500e+02 9.00e+00 2.67e+00   0.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  1.6881766e+02 7.90e+00 3.00e+00  -6.0 1.46e+01  -4.0 6.36e-02 1.23e-01h  1
   2  2.4765788e+02 1.44e+00 2.39e+00   0.4 4.05e+00  -4.5 4.61e-01 8.18e-01h  1
   3  2.4037967e+02 1.36e+00 2.09e+00  -0.9 6.08e+01  -1.3 8.77e-02 5.61e-02f  1
   4  2.3932635e+02 8.82e-01 4.17e+00  -1.1 5.78e+00  -1.8 1.34e-01 3.49e-01h  1
   5  2.4058508e+02 5.94e-01 2.85e+00  -1.9 2.08e+00  -2.3 6.35e-01 3.27e-01h  1
   6  2.3850071e+02 3.49e-01 2.45e+00  -6.6 5.01e+00  -2.8 3.07e-03 4.12e-01h  1
   7  2.4183230e+02 1.17e-02 1.84e-01  -1.2 2.34e-01  -3.3 1.00e+00 9.66e-01h  1
   8  2.4180281e+02 2.53e-06 2.48e-04  -3.0 5.00e-02  -3.7 9.98e-01 1.00e+00h  1
```

```
   9  2.4180001e+02 4.97e-09 2.36e-07  -9.0 4.96e-04  -4.2 9.98e-01 9.98e-01h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  10  2.4180000e+02 3.55e-15 1.82e-11 -11.0 8.74e-07  -4.7 1.00e+00 1.00e+00h  1

Number of Iterations....: 10
                                   (scaled)                 (unscaled)
Objective...............:   2.4179999978554039e+02    2.4179999978554039e+02
Dual infeasibility......:   1.8188284627939566e-11    1.8188284627939566e-11
Constraint violation....:   3.5527136788005009e-15    3.5527136788005009e-15
Complementarity.........:   1.1361390269409665e-11    1.1361390269409665e-11
Overall NLP error.......:   1.8188284627939566e-11    1.8188284627939566e-11

Number of objective function evaluations             = 11
Number of objective gradient evaluations             = 11
Number of equality constraint evaluations            = 11
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 11
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 10
Total CPU secs in IPOPT (w/o function evaluations)   =      0.006
Total CPU secs in NLP function evaluations           =      0.001

EXIT: Optimal Solution Found.

 The solution was found.

 The final value of the objective function is    241.799999785540

 --------------------------------------------------
 Solver         :  IPOPT (v3.12)
 Solution time  :   1.449999999022111E-002 sec
 Objective      :    241.800000645399
 Successful solution
 --------------------------------------------------

In [12]:  m.options.SOLVER=1
In [13]:  print('')
          print('Results')
          print('z11: ' + str(z11.value))
          print('z12: ' + str(z12.value))
          print('z13: ' + str(z13.value))
          print('z21: ' + str(z21.value))
          print('z22: ' + str(z22.value))
          print('z23: ' + str(z23.value))
          print('z31: ' + str(z31.value))
          print('z32: ' + str(z32.value))
          print('z33: ' + str(z33.value))
          Results
          z11: [0.0]
          z12: [13.00000004]
          z13: [0.0]
          z21: [0.0]
          z22: [5.00000001]
          z23: [8.00000001]
          z31: [11.00000003]
          z32: [0.0]
          z33: [4.00000002]
```

Examples2, 3 and 4 were extracted from Osuji et al (2014),Abdul-Salam (2014) and Opara et al. (2015) respectively, solved via the same procedure as displayed in Example 1, and their results are presented in Table 2, along with the results of others.

Table 2: Summary of Results for the Four Practical Examples Used

Example 1		
	Wolfram Mathematica Optimal Solution	$z_{12}=13,\ z_{22}=5,\ z_{23}=8,\ z_{31}=11,\ z_{33}=4$
	Anaconda Python Optimal Solution	$z_{12}=13,\ z_{22}=5,\ z_{23}=8,\ z_{31}=11,\ z_{33}=4$
	Optimal Value	**253,000**
Example 2	Wolfram Mathematica Optimal Solution	$z_{14}=5,\ z_{15}=6,\ z_{22}=7,\ z_{23}=9,\ z_{25}=1,\ \ z_{31}=6,\ z_{34}=5$
	Anaconda Python Optimal Solution	$z_{14}=5,\ z_{15}=6,\ z_{22}=7,\ z_{23}=9,\ z_{25}=1,\ \ z_{31}=6,\ z_{34}=5$
	Optimal Value	**377,000**
Example 3	Wolfram Mathematica Optimal Solution	$z_{12}=7, z_{13}=8, z_{21}=10, z_{22}=3, z_{24}=12, z_{31}=10$
	Anaconda Python Optimal Solution	$z_{12}=7, z_{13}=8, z_{21}=10, z_{22}=3, z_{24}=12, z_{31}=10$
	Optimal Value	**236,000**
Example 4	Wolfram Mathematica Optimal Solution	$z_{12}=8, z_{15}=5, z_{23}=12, z_{25}=4, z_{31}=9, z_{32}=2, z_{41}=1, z_{44}=14$
	Anaconda Python Optimal Solution	$z_{12}=8, z_{15}=5, z_{23}=12, z_{25}=4, z_{31}=9, z_{32}=2, z_{41}=1, z_{44}=14$
	Optimal Value	**509,000**

Comment [PR15]: Beware of table exceeding margins.

6 Conclusion

The study employed computer programming languages to solve transportation problem in a concave function. Four real life examples extracted from different authors of published works were employed successfully to demonstrate the effectiveness of the written programmes. Theoutput from the two programming languages is the same for optimal solution.

Comment [PR16]: I must disagree with this conclusion. The paper does not lay a logical track from the problem to be solved to the presented solution. Nor does it relate aspects of the problem to the solution method or tell what parameter values yield the best solution. Please see comments above for details.

References

Abdul-Salam, S. M. (2014). Transportation with volume discount: A case study of a logistic operator in Ghana. *Journal of Transport Literature*, *8*(2), 7–37.

Amaravathy, A., Thiagarajan, K. & Vimala, S. (2018). Optimal solution of OFSTF, MDMA methods with existing methods comparison. *International Journal of Pure and Applied Mathematics*, *119*(10), 989 –1000.

Haruna, I., Ahmed, M. & Akweittey, E. (2012). The network transportation problem with volume discount on shipping cost. *International Journal of Science and Research*, *3*(12), 1841 – 1843.

Kaur, A. & Kumar, A. (2011). A new method for solving fuzzy transportation problems using ranking function. Applied Mathematical Modelling, 35(2011), 5652–5661.

Minken, H. & Johansen, B. G. (2019). A logistics cost function with explicit transport costs. Economics of Transportation, 19(2019), 23 – 31.

Mostafa, S.A., Juman, Z. A. M. S., Nawi, N. M., Mahdin, H. & Mohammed, M.A. (2022). Improving genetic algorithm to attain better routing solutions for real-world water line system. *In International Conference on Soft Computing and Data Mining*, Switzerland, 292–301.

Okenwe, I. (2018). Solution to nonlinear transportation problem: A case study of Unilever Nigeria Plc. *International Journal of Mathematics*, *1(11)*, 1–12.

Oliveira, C., Qassim, R. & Nzualo, T. (2020). A comparative study of linear programming and nonlinear programming models of the ship speed optimization problem in maritime transportation. *Authorea, 6(15),* 46-58.

Opara, J., Iheagwara, A. I. & Nwobi, A. C. (2015). Nonlinear transportation problems' algorithm: A case study of the Nigerian bottling company ltd Owerri plant. *International Journal of Research, 2(2),* 1350–1363.

Osuji, G. A., Opara, J. & Chukwudi, J. O. (2014). Transportation algorithm with volume discount on distribution cost: A case study of the Nigerian bottling company plc Owerri plant. *American Journal of Applied Mathematics and Statistics, 2(5),* 318–323.

Psraftis, H. N. & Kontovas, C.A. (2013). Speed models for energy-efficient maritime transportation: A taxonomy and survey.*Transportation Research,26(2013),* 331–351.

Rudi, A., Fröhling, M., Zimmer, K. & Schultmann, F. (2016). Freight transportation planning considering carbon emissions and in-transit holding costs: a capacitated multi-commodity network flow model.*EURO Journal on Transportation and Logististic*, *5*(2), 123-160.