Scientific Research Publishing

Open Journal of Optimization, 2026, 15(3), 65-79
https://www.scirp.org/journal/ojop
ISSN Online: 2325-7091
ISSN Print: 2325-7105

Further Numerical Results for a New Lagrange Method for Solving Convex Quadratic Optimization Problems

John Avoka[1], Stephen B. Twum[2], Christian J. Etwire[2]

[1]Catholic University of Ghana, Fiapre-Sunyani, Ghana
[2]University of Technology and Applied Sciences, Navrongo, Ghana
Email: javoka08@gmail.com, stwum@cktutas.edu.gh, cetwire@cktutas.edu.gh

How to cite this paper: Avoka, J., Twum, S.B. and Etwire, C.J. (2026) Further Numerical Results for a New Lagrange Method for Solving Convex Quadratic Optimization Problems. *Open Journal of Optimization*, **15**, 65-79.
https://doi.org/10.4236/ojop.2026.153004

Received: July 22, 2026
Accepted: September 27, 2026
Published: September 30, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

The work reported in this paper is a sequel to an earlier paper of the authors in which a modified classical Lagrange Multiplier method was used to solve merely small-sized convex quadratic optimization problems, where hand computation was possible. In the current work, the authors demonstrate the utility of the new method in solving medium to large sized convex quadratic optimization problems for which hand computations are impossible, or cumbersome. The modified version of the Lagrange method was numerically analyzed against the classical Lagrange method using performance measures of: solution output, run-time, number of iterations needed to converge, and number of function evaluations in solving a given convex quadratic optimization problem. The hypothetically constructed problems which were run in MATLAB produced results that showed that, generally, the modified version of the Lagrange method was as good as, or out-performed, the latter in the performance indicators.

Keywords

Convex Quadratic Programming, Lagrangian Function, Lagrange Multipliers, Optimality Condition, Subsidiary Constraint Equations

1. Introduction

Optimization problems arise naturally in real life situations, as alluded to by Leonhard Euler [1]; it is thus no wonder that the study of Classical Optimization has grown rapidly over the last sixty years, especially due to the advent of the electronic computer and software development, just as its related counterpart of Lin-

ear Optimization has also grown. These have enabled numerical experimentations with solution algorithms using large scale problems, which otherwise was impossible to do. This in turn has spurred on the quest to solve large-scale real-world problems with consequent positive impact on the quality and standard of living of humanity.

Classical optimization basically refers to Nonlinear Optimization [2], which pre-dates Linear Optimization which gained impetus for rapid and revolutionary growth during the Second World War period and after [3]. Many specializations and classifications have evolved from nonlinear optimization, also called Nonlinear Programming (NP); notable amongst which is Convex Optimization (also known as Convex Programming (CP)), which has gained a lot of interest in recent times, with applications discovered in many fields including computer science, robotics and signal processing [4]. The current work is situated within a subclass of problems in Convex Optimization, known as Convex Quadratic Programming (CQP). Although they belong to the universal class of NP problems, due to their special properties [4], special solution algorithms or methods have been developed for solving them more efficiently, among which are the interior point methods.

Just as NP problems, CQP problems are also solvable by the classical Lagrange Multiplier method, for equality constrained types, and the Karush-Kuhn-Tucker (KKT) method, for both equality and mixed inequality-equality types of problems [5]. The work reported in this paper investigates further a modified version of the classical Lagrange Multiplier method, earlier developed by the authors (see [6]), in terms of performance at solving large scale CQP problems, against that of the classical Lagrange method. CQP problems are characterized by either a linear or convex quadratic objective function and linear or convex quadratic constraints [7]. Typically, it may be denoted by:

$$\begin{aligned} &\text{minimize } f(x) = c^{\mathrm{T}}x + x^{\mathrm{T}}Qx + q \\ &\text{subject to } Ax = b \end{aligned} \tag{1.1}$$

where x is a vector of n decision variables, c is n dimensional constant vector, Q and A are respectively $n \times n$ symmetric positive semi-definite and $m \times n$ constant matrices, and b is m dimensional constant vector [8]. If the objective function is linear and constraints a mixture of linear, nonlinear, convex quadratic equations, the problem becomes:

$$\begin{aligned} &\text{minimize } f(x) = c^{\mathrm{T}}x \\ &\text{subject to } h_i(x) = b_i,\ i = 1, 2, \cdots, m \end{aligned} \tag{1.2}$$

where, $h_i(x)$ is linear, or convex quadratic function, for some $i \in \{1, 2, \cdots, m\}$. A brief review of the classical Lagrange method (CLM) applied to CQP problems is necessary therefore.

Consider a CQP problem in the form:

$$\begin{aligned} &\min f(x) \\ &\text{subject to}: h_i(x) = b_i,\ b_i \in R,\ x \in R^n,\ i = 1, \cdots, m \end{aligned}$$

where f and h_i are respectively linear objective function and convex quadratic or linear i^{th} constraint function, for some i. Putting $g_i(x) = h_i(x) - b_i = 0$, the problem may be posed as:

$$\begin{aligned} &\min f(x) \\ &\text{subject to}: g_i(x) = 0,\ x, b_i \in R^n,\ i = 1, \cdots, m. \end{aligned} \tag{1.3}$$

The CLM constructs a Lagrangian function [7] from problem (1.3) as:

$$\min L(x, \lambda) = f(x) - \lambda^{\mathrm{T}} g(x) \tag{1.4}$$

where L is a Lagrangian function, $g(x) = \left(g_1(x), g_2(x), \cdots, g_m(x)\right)^{\mathrm{T}}$ is a vector of constraint functions and $\lambda = \left(\lambda_1, \lambda_2, \cdots, \lambda_m\right)^{\mathrm{T}}$ is a vector of m unknown multipliers (scalars). Therefore, the equality constrained problem (1.3), by the construction of the Lagrangian function, is transformed into an unconstrained problem to which the first order necessary optimality conditions of unconstrained optimization can be applied, leading to the equations:

$$\nabla_x L(x, \lambda) = \nabla_x f(x) - \lambda^{\mathrm{T}} \nabla_x g(x) = 0 \tag{1.5}$$

$$\nabla_\lambda L(x, \lambda) = \nabla_\lambda f(x) - g(x) = 0 \tag{1.6}$$

For convex problems the conditions (1.5) and (1.6) are both necessary and sufficient for a minimum solution of (1.4) and so of (1.3) [9]. Therefore, if there exists a solution x^*, λ^* to (1.5) and (1.6), then it is a solution to (1.4) also, and vice versa. From the equations (1.5) and (1.6), we obtain a system of $m + n$ equations which are solved simultaneously to furnish the solution. High computational effort and time is required when m and n are very large. A method that obtains the minimum solution with relatively less effort and in a shorter time, therefore, becomes desirable. This is the motivation for the current work.

For some previous related works involving the Lagrange method, readers are referred to [10] [11], Fortin and [12]-[17]. The next two sections respectively present a review of the modified Lagrange method (MLM) as presented in [6], and the results from applications of the method to some hypothetical problems. The paper concludes with some highlights from the MLM and future prospects for it.

2. A Review of the MLM

The MLM is premised on the grounds that instead of simultaneous solution of the problems (1.5) and (1.6), as the case is in CLM, the solution x^* can be obtained independently from (1.5) alone, and thereafter the solution λ^* can also be obtained from the same (1.5). The MLM, therefore effectively departs from the CLM by decomposing the solution process into two independent ones, where the entire solution is obtained solely on the basis of (1.5). For more details the reader is referred to [6].

The scalar form of Equation (1.5) is:

$$\frac{\partial f(x)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(x)}{\partial x_j} = 0,\ i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n. \tag{2.1}$$

We observe that (2.1) is a system of n equations in n variables of x and m variables of λ. Since $g_i(x)$ and $\frac{\partial g_i(x)}{\partial x_j}$ both identically vanish at x^* for any i and j, in spite of λ_i [6] it means λ_i is arbitrary for all i, as far as finding x^* from (2.1) is concerned, and so λ_i can be chosen arbitrarily [6]. Choosing $\lambda_i \neq 0$ for some $i \in \{1,2,\cdots,m\}$ and $\lambda_s = 0$ for all $s \neq i$, $s \in \{1,2,\cdots,m\}$, we obtain the result:

$$\frac{\partial f(x)}{\partial x_j} = \lambda_i \frac{\partial g_i(x)}{\partial x_j},\ j = 1,2,\cdots,n,\ \forall i \in \{1,2,\cdots,m\} \quad [6] \tag{2.2}$$

From (2.2), λ_i can be eliminated by taking ratios involving ordered pairs of $j = 1,2,\cdots,n$ (see [6]). For instance, by taking ratios of the j^{th} and the $(j+1)^{\text{th}}$ equations $(j = 1,2,\cdots,n-1)$ we obtain the equations:

$$\frac{\partial f(x)/\partial x_1}{\partial f(x)/\partial x_2} = \frac{\partial g_i(x)/\partial x_1}{\partial g_i(x)/\partial x_2}, \frac{\partial f(x)/\partial x_2}{\partial f(x)/\partial x_3} = \frac{\partial g_i(x)/\partial x_2}{\partial g_i(x)/\partial x_3}, \cdots,$$
$$\frac{\partial f(x)/\partial x_{n-1}}{\partial f(x)/\partial x_n} = \frac{\partial g_i(x)/\partial x_{n-1}}{\partial g_i(x)/\partial x_n}$$

which are generalized as:

$$\frac{\partial f(x)/\partial x_j}{\partial f(x)/\partial x_{j+1}} = \frac{\partial g_i(x)/\partial x_j}{\partial g_i(x)/\partial x_{j+1}},\ j = 1,2,\cdots,n-1,\ \forall i \in \{1,2,\cdots,m\}$$

leading to:

$$\frac{\partial f(x)}{\partial x_j}\frac{\partial g_i(x)}{\partial x_{j+1}} - \frac{\partial f(x)}{\partial x_{j+1}}\frac{\partial g_i(x)}{\partial x_j} = 0,\ j = 1,2,\cdots,n-1;\ \forall i \in \{1,2,\cdots,m\} \tag{2.3}$$

The Equation (2.3) are called Subsidiary Constraint Equations (SCE), as in [6], and for each i, they produce $n-1$ equations. For CQP problems, the SCE are linear. The SCE therefore become additional available equations to the m original ones from $g_i(x) = 0$, yielding $m+n-1$ equations in x only and which generally are simpler to solve for x^* in the kind of problem studied. As observed in [6], some of the equations in (2.3) can be redundant. Different forms of ordering of the ratios have been found to improve the situation and address the issue of redundant equations. For instance, taking consecutive ratios between the first equation and subsequent ones, yield:

$$\frac{\partial f(x)/\partial x_j}{\partial f(x)/\partial x_{j+1}} = \frac{\partial g_i(x)/\partial x_j}{\partial g_i(x)/\partial x_{j+1}}, \frac{\partial f(x)/\partial x_j}{\partial f(x)/\partial x_{j+2}} = \frac{\partial g_i(x)/\partial x_j}{\partial g_i(x)/\partial x_{j+2}}, \cdots,$$
$$\frac{\partial f(x)/\partial x_j}{\partial f(x)/\partial x_n} = \frac{\partial g_i(x)/\partial x_j}{\partial g_i(x)/\partial x_n}$$

where $j \in \{1,2,\cdots,n-1\}$ and $i \in \{1,2,\cdots,m\}$. This is generalized as:

$$\frac{\partial f(x)/\partial x_j}{\partial f(x)/\partial x_{j+s}} = \frac{\partial g_i(x)/\partial x_j}{\partial g_i(x)/\partial x_{j+s}},\ j \in \{1,2,\cdots,n-1\},\ s = 1,2,\cdots,n-1,\ \forall i$$

It follows that:

$$\frac{\partial f(x)}{\partial x_j}\frac{\partial g_i(x)}{\partial x_{j+s}}-\frac{\partial f(x)}{\partial x_{j+s}}\frac{\partial g_i(x)}{\partial x_j}=0,\ j\in\{1,2,\cdots,n-1\},\ s=1,2,\cdots,n-1,\ \forall i \tag{2.4}$$

Also, taking aggregates of the SCE as in (2.4) tend to be more effective against producing redundant equations [6]. Summing therefore SCE of (2.4) for $k=j\geq 1$, involving $k,k+1,\cdots,n-1,n$, we obtain:

$$\frac{\partial f(x)}{\partial x_k}\left(\frac{\partial g_i(x)}{\partial x_{k+1}}+\frac{\partial g_i(x)}{\partial x_{k+2}}+\cdots+\frac{\partial g_i(x)}{\partial x_n}\right)$$
$$-\left(\frac{\partial f(x)}{\partial x_{k+1}}+\frac{\partial f(x)}{\partial x_{k+2}}+\cdots+\frac{\partial f(x)}{\partial x_n}\right)\frac{\partial g_i(x)}{\partial x_k}=0,$$

which is the same as:

$$\frac{\partial f(x)}{\partial x_k}\sum_{s=k+1}^{n}\frac{\partial g_i(x)}{\partial x_s}-\frac{\partial g_i(x)}{\partial x_k}\sum_{s=k+1}^{n}\frac{\partial f(x)}{\partial x_s}=0,1\leq k\leq n-2 \tag{2.5}$$

From the SCE (2.4) or (2.5), it is noted that various sums involving two (2) or more SCE can be produced. This is indication that a variety of SCE derived from aggregating some other SCE can be created as in (2.5). Thereafter, the resulting SCE, together with the constraint equations $g_i(x)=0$ can be solved simultaneously to find independently the minimum solution x^*; the corresponding minimum objective function value $f(x^*)$ can therefore be obtained. Furthermore, the optimal value λ^* can be independently obtained from the solution of the system of equations:

$$\frac{\partial f(x^*)}{\partial x_j}-\sum_{i=1}^{m}\lambda_i\frac{\partial g_i(x^*)}{\partial x_j}=0,\ i=1,2,\cdots,m;\ j=1,2,\cdots,n$$

which involves the variables λ_i only, $i=1,2,\cdots,m$.

3. Numerical Results

This section is divided into two: The first compares the performance of the MLM against the CLM in connection with solving small-size problems; the second compares the MLM against the CLM for medium-to-large-size problems. Three (3) small-size problems and four (4) medium-to-large problems were run in MATLAB and their run times, objective function values, number of iterations, the solutions obtained (including values of the multipliers), and number of function evaluations compared. The problems considered in this paper are limited to linear equality constraints.

The results are presented in Tables 1-3 for the small-size problems and in Tables 4-7 for the medium-to-large problems. The *Sequential Quadratic Programming* algorithm in MATLAB was used in running the problems with some modified codes incorporated in MATLAB for running the MLM (see the **Appendix**). The termination tolerance on the function value in MATLAB was 1e−20; the tolerance on the constraint violation was 1e−06; the maximum function evaluations

were 100*number of variables; maximum number of iterations allowed was 400; Termination tolerance on the step size was taken to be 1e−06; Termination tolerance on the first-order optimality is taken as 1e−06.

3.1. Small-Size Problems

The problems are taken from [18]. Problem P1 consists of five (5) decision variables and two (2) constraints, Problem P2 consists of five (5) decision variables and three (3) constraints and Problem P3 consists of five (5) decision variables and three (3) constraints. The results are displayed in Tables 1-3.

Table 1. Performance of MLM against CLM in Problem P1.

	CLM	MLM
Initial values	$[0, 0, 0, 0, 0]^T$	$[0, 0, 0, 0, 0]^T$
Number of iterations	9	3
Run time (Sec)	0.0340	0.0330
Objective function value	2.1012e−16	21012e−16
Solutions (x^*)	$[1, 1, 1, 1, 1]^T$	$[1, 1, 1, 1, 1]^T$
Solutions (λ^*)	$[-0.3947e-07, 0.0510e-07]^T$	$[-0.3947e-07, 0.0510e-07]^T$
Function Evaluations	50	18

Table 2. Performance of MLM against CLM in Problem P2.

	CLM	MLM
Initial values	$[0, 0, 0, 0, 0]^T$	$[0, 0, 0, 0, 0]^T$
Number of iterations	6	6
Run time (s)	0.0216	0.0204
Objective function values	5.3266	5.3266
Solution (x^*)	$[-0.0946, 0.0315, 0.5158, -0.4527, 0.0315]^T$	$[-0.0946, 0.0315, 0.5158, -0.4527, 0.0315]^T$
Solution (λ^*)	$[3.2779, 2.9054, 7.7479]^T$	$[3.2779, 2.9054, 7.7479]^T$
Function Evaluations	50	50

Table 3. Performance of MLM against CLM in Problem P3.

	CLM	MLM
Initial values	$[0, 0, 0, 0, 0]^T$	$[0, 0, 0, 0, 0]^T$
Number of iterations	6	2
Run time (s)	0.0257	0.0243
Objective function value	4.0930	4.0930
Solution (x^*)	$[-0.7674, 0.2558, 0.6279, -0.1163, 0.2558]^T$	$[-0.7674, 0.2558, 0.6279, -0.1163, 0.2558]^T$

Continued

Solution (λ^*)	$[2.0465, 2.2326, -5.9533]^T$	$[2.0465, 2.2326, -5.9533]^T$
Function Evaluations	44	40

3.2. Medium to Large-Size Problems

The four (4) problems P4 to P7 are hypothetically constructed CQP problems, generally of the form:

$$\min f(x) = x^T Qx + c^T x + q$$
$$\text{subject to } Ax = b$$

where $Q \in R^{n\times n}$, $x, c \in R^n$, $q \in R$, $A \in R^{m\times n}$, $b \in R^m$.

Problem P4 has $n = 50$ and $m = 15$. Q is composed of entries 9 on its main diagonal and 7 on its off-diagonals; c has 1 for each of its components and q is 1. The matrix A is extracted as a submatrix of size 15 × 50 from a 50 × 50 identity matrix. Each component of the vector b is 1, formed by summing up the corresponding elements of each row of A.

Problem P5 has $n = 200$ and $m = 100$. Q has entries $\frac{229}{2}$ on its main diagonal and $\frac{225}{2}$ on its off-diagonal; c has 1 for each of its components and q is 2.5. The matrix A is extracted as a submatrix of size 100 × 200 from a tridiagonal matrix with entries 2, 1 and 1 of size 200 × 200. The components of the vector b are formed by summing up the corresponding elements of each row of A.

Problem P6 has $n = 500$ and $m = 30$. Q has16 on its main diagonal entries and 7 on its off-diagonal entries. c has 1 as components and q is 1. The matrix A is of size 30 × 500 with 1 as its elements. The components of the vector b are the sums of the corresponding elements of each row of A.

Problem P7 has $n = 200$, and $m = 100$. Q has $\frac{49}{2}$ on its main diagonal and $\frac{45}{2}$ on its off-diagonal; each component of c is 1 and the value of q is 10. The matrix A is 100 × 200 with 1 as its entries. The components of the vector b are formed by summing up the corresponding elements of each row of A.

Tables 4-7 present the computational results obtained after solving the problems in MATLAB.

Table 4. Performance of the MLM against CLM in Problem P4.

	CLM	MLM
Initial value	$0 \in R^{50}$	$0 \in R^{50}$
Number of iterations	5	5
Run time (s)	11.2429	10.1382
Objective function value	22.4271	22.4271

Continued

Solution (x^*)	Same as CLM	Same as MLM
Solution (λ^*)	Same as CLM	Same as MLM
Function Evaluations	319	319

Table 5. Performance of MLM against CLM in Problem P5.

	CLM	MLM
Initial value	$0 \in R^{200}$	$0 \in R^{200}$
Number of iterations	7	9
Run time (s)	35.6007	33.1118
Objective function value	197.0847	197.0847
Solution (x^*)	Did not converge	Converged to solution
Solution (λ^*)`	No solution	Solution obtained
Function Evaluations	3051	2027

Table 6. Performance of MLM against CLM in Problem P6.

	CLM	MLM
Initial value	$0 \in R^{500}$	$0 \in R^{500}$
Number of iterations	6	4
Run time (s)	18.3523	17.0087
Objective function value	128.2319	134.7099
Solution (x^*)	Did not converge	Converged to solution
Solution (λ^*)	No solution	Solution obtained
Function Evaluations	3539	2013

Table 7. Performance of MLM against CLM in Problem P7.

	CLM	MLM
Initial values	$0 \in R^{200}$	$0 \in R^{200}$
Number of iterations	4	2
Run time (s)	0.0876	0.0639
Objective function value	212.0049	212.0049
Solution (x^*)	Same as MLM	Same as CLM
Solution (λ^*)	Same as MLM	Same as CLM
Function Evaluations	6028	2268

To give a quick glance at all the results presented in Tables 1-7, a composite

table (Table 8) is constructed. The following keys are used in Table 8 as headings: Initial Value (IV), Number of Iterations (NIT), Run-Time (RT), Objective Function Value (OFV), Solution (SLN), and Number of Function Evaluations (NFE).

Table 8. Composite table of results for problems P1 to P7.

Problem	Method	IV	NIT	RT (s)	OFV	SLN (x^*)	SLN (λ^*)	NFE
P1	CLM	$0 \in R^5$	9	0.0340	2.1012e−16	Same as MLM	Same as MLM	50
	MLM	$0 \in R^5$	3	0.0330	2.1012e−16	Same as CLM	Same as CLM	18
P2	CLM	$0 \in R^5$	6	0.0216	5.3266	Same as MLM	Same as MLM	50
	MLM	$0 \in R^5$	6	0.0204	5.3266	Same as CLM	Same as CLM	50
P3	CLM	$0 \in R^5$	6	0.0257	4.0930	Same as MLM	Same as MLM	44
	MLM	$0 \in R^5$	2	0.0243	4.0930	Same as CLM	Same as CLM	40
P4	CLM	$0 \in R^{50}$	5	11.2429	22.4271	Same as MLM	Same as MLM	319
	MLM	$0 \in R^{50}$	5	10.1382	22.4271	Same as CLM	Same as CLM	319
P5	CLM	$0 \in R^{200}$	7	35.6007	197.0847	Did not converge	No solution	3051
	MLM	$0 \in R^{200}$	9	33.1118	197.0847	Converged	Converged	2027
P6	CLM	$0 \in R^{500}$	6	18.3523	128.2319	Did not converge	No solution	3539
	MLM	$0 \in R^{500}$	4	17.0089	134.7099	Converged	Converged	2013
P7	CLM	$0 \in R^{200}$	4	0.0876	212.0049	Same as MLM	Same as MLM	6028
	MLM	$0 \in R^{200}$	2	0.0639	212.0049	Same as CLM	Same as CLM	2268

3.3. Analysis of Residual Breakdown under Algorithmic Failure

Table 9 shows the breakdown of P5 - P7, stating the reasons why those problems failed to converge.

To fully understand the mathematical breakdown that occurs when the Classical Lagrange Method (CLM) deals with large-scale or rank-deficient systems, we analyze the final states of Problem P5, Problem P6, and Problem P7.

When a solver fails to converge, its performance is measured by how much it violates the first-order optimality condition (gradient alignment) and the feasibility condition (constraint boundaries). These are quantified using infinity-norm metrics:

1. Optimality Residual (R_{opt}): $\left\|\nabla f\left(x^*\right)+A^{\mathrm{T}}\lambda^*\right\|_\infty$
2. Feasibility residuals (R_{feas}): $\left\|Ax^*-b\right\|_\infty$

P5 failed to converge under CLM due to Saddle-Point Ill-Conditioning: The large variable space caused standard KKT matrix iterations to lose tracking and drift completely off the feasible hyperplane.

P6, also failed to converge under CLM due to Total Rank Deficiency: A matrix full of uniform ones creates a rank of 1. The CLM's internal matrix inverse col-

lapsed, causing exploded gradients.

Table 9. Residuals and Exit flags of Problems P5 - P7.

Problem	Method	Optimality Residual	Feasibility Residual	Exit Flag
P5	CLM	1.8432×10^{2}	4.5019×10^{-1}	−2 (Diverged)
	MLM	7.1804×10^{-9}	1.7043×10^{-11}	1
P6	CLM	4.7690×10^{4}	3.1250×10^{0}	−5 (Singular)
	MLM	1.3303×10^{-10}	2.1794×10^{-12}	1
P7	CLM	8.9214×10^{-2}	1.2045×10^{-3}	4 (Stalled)
	MLM	5.6809×10^{-10}	8.810×10^{-12}	1

Finally, P7 failed to also converge under CLM due to Boundary Stalling: The CLM solver stopped prematurely due to a pathologically small step size, failing to reach standard scientific precision limits.

4. Discussions and Conclusions

For the small-size problems (see Tables 1-3, or Table 8), the MLM out-performed the CLM in all the performance measures, or was as good. The MLM produced the same solutions as the CLM as far as the objective function values and the values of x^* and λ^* were concerned. In terms of number of iterations, the MLM did better in two of them (*i.e.* P1 and P3) and did as well as the CLM in P2. In terms of run time, again the MLM was better than the CLM in all three problems, even though the difference tended to be marginal, something that can be attributed to the size of the problem (see the run-times for the medium to large problems, which show significant difference). The MLM performed better than the CLM in terms of the number of function evaluations in all three problems except P2, where it had equal number function evaluations as the CLM.

For the medium to large problems (P4 to P7), the same outcomes were achieved in all the performance indicators, except that in Problems P5 and P6, while the CLM failed to converge and thus did not yield solutions, the MLM converged.

The results obtained for all seven problems indicate the superiority of the MLM over the CLM in solving CQP problems of the type specified in this paper. Future work would seek to explore the performance of the MLM in solving other forms of the CQP problem, including solving convex problems in general. Other future works would explore extension of the methodology of the MLM to solve KKT-like CQP problems, where inequality constraints are involved. It is anticipated too that the MLM could have far-reaching consequence for efficient solution of CQP problems using interior point methods.

Author Contributions

Conceptualization, John Avoka and Stephen B. Twum; methodology, John Avoka

and Stephen B. Twum; software, John Avoka; validation, John Avoka, Stephen B. Twum, and Christian J. Etwire; formal analysis, Stephen B. Twum; investigation, Stephen B. Twum; resources, John Avoka; data curation, John Avoka; writing—original draft preparation, John Avoka; writing—review and editing, Christian J. Etwire; visualization, Christian J. Etwire; supervision, Stephen B. Twum; project administration, Christian J. Etwire. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Yan, Y.M. (2015) Active-Set Prediction for Interior Point Methods. Ph.D. Thesis, University of Edinburgh.

[2] Bazaraa, M.S., Sherali, H.D. and Shetty, C.M. (2006) Nonlinear Programming: Theory and Algorithms. John Wiley & Sons.

[3] Dantzig, G.B. (1951) Maximization of a Linear Function of Variables Subject to Linear Inequalities. In: Koopmans, K.C., Ed., *Activity Analysis of Production and Allocation*, John Wiley & Sons, 339-347. https://www.scirp.org/reference/referencespapers?referenceid=610257

[4] Boyd, S. and Vandenberghe, L. (2004) Convex Optimization. Cambridge University Press.

[5] Hoffman, L.D. and Bradley, G.L. (1992) Calculus for Business, Economics, and the Social and Life Sciences. McGraw-Hill.

[6] Twum, B.S., Christian, J.E. and John, A. (2024) A Modified Lagrange Method for Solving Convex Quadratic Optimization Problems. *Open Journal of Optimization*, **13**, 1-20. https://www.scirp.org/journal/paperinformation?paperid=131813

[7] Nocedal, J. and Wright, S.J. (2006) Quadratic Programming. In: Nocedal, J. and Wright, S.J., Eds., *Numerical Optimization*, Springer, 448-496. https://doi.org/10.1007/978-0-387-40065-5_16

[8] Shustrova, A. (2015) Primal-Dual Interior Methods for Quadratic Programming. Master's Thesis, University of California. https://escholarship.org/content/qt15n5764p/qt15n5764p.pdf?v=lg

[9] Moura, S. (2018) Renewable Energy Systems and Control: KKT Conditions and Duality Theory [Lecture Notes]. University of California.

[10] Hestenes, M.R. (1969) Multiplier and Gradient Methods. *Journal of Optimization Theory and Applications*, **4**, 303-320. https://doi.org/10.1007/bf00927673

[11] Powell, M.J. (1969) A Method for Nonlinear Constraints in Minimization Problems. In: Fletcher, R., Ed., *Optimization*, Academic Press, 283-298.

[12] Fortin, M. and Glowinski, R. (1983) Augmented Lagrangian Methods: Numerical Solution of Boundary-Value Problems. Elsevier.

[13] Komzsik, L. and Chiang, K. (1993) The Effect of a Lagrange Multiplier Approach in MSC/NASTRAN on Large-Scale Parallel Applications. *Computing Systems in Engineering*, **4**, 399-403. https://doi.org/10.1016/0956-0521(93)90008-k

[14] Bertsekas, D.P. (1996) Constrained Optimization and Lagrange Multiplier Methods. Athena Scientific.

https://www.scirp.org/reference/referencespapers?referenceid=1976048

[15] Liang, X., Hu, J., Zhong, W. and Qian, J. (2008) A Modified Augmented Lagrange Multiplier Method for Large-Scale Optimization. *Developments in Chemical Engineering and Mineral Processing*, **9**, 115-124. https://doi.org/10.1002/apj.5500090214

[16] Curtis, F.E., Jiang, H. and Robinson, D.P. (2015) An Adaptive Augmented Lagrangian Method for Large-Scale Constrained Optimization. *Mathematical Programming*, **152**, 201-245. https://doi.org/10.1007/s10107-014-0784-y

[17] Burman, E., Hansbo, P. and Larson, M.G. (2023) The Augmented Lagrangian Method as a Framework for Stabilised Methods in Computational Mechanics. *Archives of Computational Methods in Engineering*, **30**, 2579-2604. https://doi.org/10.1007/s11831-022-09878-6

[18] Schittkowski, K. (2012) More Test Examples for Nonlinear Programming Codes (vol. 282). Springer Science & Business Media. https://www.scirp.org/reference/referencespapers?referenceid=3689508

Appendix. An Example of the Modified Lagrange Algorithm Codes Run on MATLAB

```
n=50
x=sym('x',[1 n])
var=(x);
i=1:n;
fun=@(x)sum(x.^2)+sum(3.5*x(i))*sum(x(i))+ sum(x(i))+1;
f= sum(x.^2)+sum(3.5*x(i))*sum(x(i))+ sum(x(i))+1
x0=zeros(1,n);
H=eye(n);
A=[];
b=[];
Aeq=H(1:15,:);
beq=sum(Aeq,2);
lb=[]
ub=[]
nonlcon=[]
options = optimoptions(@fmincon,'Algorithm','sqp','Display','off')
tic
[X,fval,exitflag,output,lambda]  =  fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,op-
tions)
toc
q=lambda.eqlin
t=nnz(q);
if t~=0
l=Aeq(t,:)
else
l=Aeq(1,:)
end
syms(sym('x',[1 n]))
g=l*x'
[gx]=gradient(g,x)
[fx]=gradient(f,x)

 K=n-2:n-1
ii=2:n

eqn1=(1/n-1)*(fx(1)*sum(gx(ii))-gx(1)*sum(fx(ii)));
 [cxy txy]=coeffs(eqn1, var)
zvar=setdiff(var,txy)
xx=symvar(eqn1)
 Coef_matrix=zeros(numel(eqn1),numel(xx));
     j=1
     [v1,v2]=coeffs(eqn1(j),xx)

     if isequal(size(v1),size(xx))
        c1=double([0])
else
    disp('the last coefficient of v1 is zero')
        c1=double([v1(end)])
     end

 if isequal(size(txy),size(xx))
xvar=setdiff(var,txy);
h1=[cxy, zeros(1,length(zvar))]
z1=double(h1)
 else
```

```
     s1=cxy(1:end-1)
    h1=[s1, zeros(1,length(zvar))]
    z1=double(h1)
 end

options = optimoptions(@fmincon,'Algorithm','sqp','Display','off');
tStart=tic
[x,fval,exitflag,output,~]=fmincon(fun,[x0],[],[],[Aeq;z1],[beq                ;-
c1],[],[],[],options)
a1=x;
m1=fval;
p1=[output.iterations];
e1=exitflag
tEnd=toc(tStart);
r1=tEnd;

eqn2=0.5*((gx(n)*sum(fx(K))-fx(n)*sum(gx(K))));
[dxy fxy]=coeffs(eqn2, var)
xvar=setdiff(var,fxy)
xx=symvar(eqn2)
 Coef_matrix=zeros(numel(eqn2),numel(xx))
     j=1
     [u1,u2]=coeffs(eqn2(j),xx)

     if isequal(size(u1),size(xx))
        c2=double([0])
else
    disp('the last coefficient of v1 is zero')
        c2=double([u1(end)])
     end

 if isequal(size(fxy),size(xx))
xvar=setdiff(var,fxy)
h2=[dxy, zeros(1,length(xvar))]
z2=double(h2)
 else
     s2=dxy(1:end-1)
    h2=[s2, zeros(1,length(xvar))]
    z2=double(h2)
 end

options = optimoptions(@fmincon,'Algorithm','sqp','Display','off');
tStart=tic;
[x,fval,exitflag,output,lambda]=fmincon(fun,[x0],[],[],[Aeq;z2],[beq;-
c2],[],[],[],options)
a2=x;
m2=fval;
p2=[output.iterations];
e2=exitflag
tEnd=toc(tStart);
r2=tEnd;

 eqn3=0.5*(eqn1+eqn2);
[bxy qxy]=coeffs(eqn3, var)
yvar=setdiff(var,qxy)
```

```
xx=symvar(eqn3)
 Coef_matrix=zeros(numel(eqn3),numel(xx))
     j=1
     [w1,w2]=coeffs(eqn3(j),xx)

     if isequal(size(w1),size(xx))
        c3=double([0])
else
    disp('the last coefficient of w1 is zero')
        c3=double([w1(end)])
     end

 if isequal(size(qxy),size(xx))
yvar=setdiff(var,qxy)
h3=[bxy, zeros(1,length(yvar))]
z3=double(h3)
 else
     s3=bxy(1:end-1)
    h3=[s3, zeros(1,length(yvar))]
    z3=double(h3)
 end

options = optimoptions(@fmincon,'Algorithm','sqp','Display','off');
tStart=tic
[x,fval,exitflag,output,lambda]=fmincon(fun,[x0],[],[],[Aeq;z3],[beq;-
c3],[],[],[],options)
a3=x;
m3=fval;
p3=[output.iterations];
e3=exitflag
tEnd=toc(tStart);
r3=tEnd;

A=[e1,e2,e3];
[minA,maxA]=bounds(A)
y1=([m1;p1;r1]);
y2=([m2;p2;r2]);
y3=([m3;p3;r3]);
if maxA == e1 && m1 < m2 && m1 < m3
    disp(y1);
elseif maxA == e2 && m2 < m1 && m2 < m3
    disp(y2);
else
    disp(y3);
end

y1=([a1]);
y2=([a2]);
y3=([a3]);
if maxA ==e1 && m1 < m2 && m1 < m3
    disp(y1);
elseif maxA==e2 && m2 < m1 && m2 < m3
    disp(y2);
else
    disp(y3);
end
```