Inequality Constrained Quadratic Programming Problems via Residual Minimization Approach

BY:

Opara, Jude[1*], Oti, Eric Uchenna[2], Osuagwu, Chidimma Udo[3] & Agwi, Uche Celestine [4]

[1,4]Department of Mathematics & Computer Science, University of Africa, Toru-Orua Bayelsa State
[2]Department of Statistics, Federal Polytechnic, Ekowe Bayelsa State
[3]Department of Statistics, Federal University of Technology, Owerri, Imo State Nigeria

Abstract

This study introduced a new and simplified technique for solving linear inequality constrained quadratic programming problems, using a regression-inspired least squares technique. The developed approach termed Inequality Constrained Quadratic Programming via Residual Minimization Approach (ICQPP-RMA) solved the first-order optimality condition $B\underline{x} + \underline{c} = \underline{0}$ *as a linear system, and when the solution is infeasible, it is projected onto the feasible region defined by the constraints. The approach avoided the complexity of traditional optimization routines such as the Karush-Kuhn-Tucker (KKT) conditions associated with the Dantzig-Wolfe algorithm, active-set, or interior-point methods. Three quadratic programming problem sets were solved manually for both Dantzig-Wolfe algorithm and ICQPP-RMA, though the Dantzig-Wolfe decomposition algorithm, and the three Mathematica in-built solvers known as standard symbolic approach (Maximize/Minimize), numerical global optimization (NMaximize/NMinimize), and local optimization techniques (FindMaximum/FindMinimum) were used as a benchmark. The new technique demonstrated to be more direct, simpler, and required fewer computational steps, even though the produced same optimal solution, which were valid. The explicit algorithms for evaluating the whole three problem set were stated in the methodology section, as well as its corresponding scripts in Wolfram Mathematica. The result when applied using Mathematica scripts with the five methods reveals that while all methods produced correct and constraint-compliant results, the ICQPP-RMA clearly stands out in terms of computational efficiency. It significantly reduces execution time compared to both standard Mathematica methods and advanced techniques like Dantzig-Wolfe, making it a promising tool for real-time or large-scale quadratic programming applications where speed and accuracy are critical. The study recommended among others that researchers and practitioners in optimization, operations research, engineering, and finance consider integrating ICQPP-RMA into their computational toolkits for solving inequality-constrained quadratic programming problems.*

Keywords**:** *Inequality Constraints*; *Quadratic Programming Problems*; *New Approach*; *Residual Minimization, Wolfram Mathematica*, *Dantzig-Wolfe algorithm*

1 Introduction

In today's increasingly complex and resource-limited world, decision-makers across industries such as finance, engineering, logistics, healthcare, and energy are often faced with the need to make optimal choices under constraints [1]. Whether it involves minimizing cost, maximizing efficiency, or balancing risk and return, many of these decisions involve trade-offs and are subject to real-world limitations such as budgets, regulations, or operational capacity. These types of problems are commonly known as constrained optimization problems [2]. Among them, Quadratic Programming (QP) problems are especially important because they allow for modeling situations where the objective (that to minimize or maximize) has a curved or nonlinear nature, such as in cases where returns diminish over time, or risks increase at an accelerating rate [3]. Quadratic programming stands apart from simpler linear programming because it can capture such relationships more realistically [4].

A particular class of interest is the inequality constrained quadratic programming problem. These are optimization problems where the objective is quadratic in nature, but the solutions must satisfy one or more inequality conditions for example, staying under a budget, meeting safety limits, or ensuring a minimum production level. These types of problems frequently arise in portfolio optimization, power grid operation, traffic control, machine learning, and even in public policy settings where fair allocation of resources is required under constraints [5]. Solving inequality constrained quadratic programming problems is challenging because the presence of inequalities introduces discontinuities and complexity in how feasible solutions are identified and improved [6]. While many classical solution techniques such as active-set methods, interior-point algorithms, and penalty-based approaches have been widely studied and applied, they can become computationally intensive, unstable, or difficult to implement in large-scale or highly sensitive problems [7].

In response to these challenges, researchers and practitioners continue to search for simpler, faster, and more reliable approaches to solving such problems, especially those that are easier to interpret or adapt to specific use cases. One emerging direction in this regard is the development of residual minimization approaches. These methods focus on measuring how far a given solution is from satisfying the optimality conditions (known as residuals) and aim to minimize this gap. By doing so, they offer a potentially more direct and intuitive path toward optimal solutions, especially in scenarios where traditional methods struggle. Despite growing interest in this area, there remains a research gap in developing and validating residual minimization techniques specifically for inequality constrained quadratic programming problems. Most existing methods focus either on unconstrained problems or equality constraints, leaving a need for tailored methods that can efficiently and accurately handle inequality conditions without excessive complexity. This study therefore aims to contribute to the field by exploring how residual-based approaches can be applied to inequality constrained quadratic programming problems. It seeks to bridge the gap between mathematical rigor and practical usability, offering

new insights and techniques that could benefit both academic research and real-world decision-making processes.

2 Materials and Methods

An optimization problem whose objective function is quadratic and constraints are linear is said to be a Quadratic Programming Problem (QPP). A QPP is defined mathematically [8] as:

$$\left.\begin{array}{l} Max.\,(or\ Min.)\ z = \underline{c}^T\,\underline{x} + \underline{x}^T D\underline{x} \\ Subject\ to: A\underline{x} \le (or \ge)\underline{b} \\ \underline{x} \ge \underline{0} \end{array}\right\} \tag{1}$$

where

$$\underline{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix}, \underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}, \underline{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}, \ A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} and\ D = \begin{pmatrix} d_{11} & d_{12} & d_{13} & \cdots & d_{1n} \\ d_{21} & d_{22} & d_{23} & \cdots & d_{2n} \\ d_{31} & d_{32} & d_{33} & \cdots & d_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{m1} & d_{m2} & d_{m3} & \cdots & d_{mn} \end{pmatrix}$$

The function $\underline{x}^T D\underline{x}$ defines a quadratic form, where D is symmetric. The matrix D is assumed negative definite if the problem is maximization and positive definite if the problem is minimization. This implies that z is strictly convex in $\underline{x}$ for minimization and strictly concave for maximization [9]. The constraints are assumed linear in this case which guarantees a convex solution space.

2.1 Inequality Constraints

The inequality constrained QPP is as presented in Equation (1). The solution to this problem is achieved by direct implantation of the Kuhn-Tucker necessary conditions. To solve Inequality Constrained Quadratic Programming Problems (ICQPP), the Dantzig-Wolfe algorithm is employed [10]. In using this technique, the Kuhn-Tucker conditions are first obtained and added as constraints to the QP and would be solved by the Simplex method. The solution of the system is obtained by using phase-1 of the Two-phase method, that is;

$$\left.\begin{array}{l} Min.\ r_0 = IR \\ Subject\ to: A\underline{x} + IR = \underline{b} \end{array}\right\} \tag{2}$$

When the sum of the artificial variables is equal to zero, Phase-1 will end in the usual manner if and only if the problem has a feasible space.

Consider the maximization problem, and the minimization one can easily be obtained. Equation (1) can be written as;

$$\left.\begin{aligned} &Max.\ z = \underline{c}^T \underline{x} + \underline{x}^T D\underline{x} \\ &Subject\ to: G(\underline{x}) = \begin{pmatrix} A \\ -1 \end{pmatrix}\underline{x} = \begin{pmatrix} \underline{b} \\ \underline{0} \end{pmatrix} \end{aligned}\right\} \tag{3}$$

Let $\underline{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_m \end{pmatrix}$ and $\underline{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \mu_m \end{pmatrix}$ be the constants in the Kuhn-Tucker conditions. Applying the Kuhn-Tucker conditions to obtain Equation (4), that is

$$\left(\begin{array}{c|c|c|c} -2D & A^T & -I & \underline{0} \\ \hline A & \underline{0} & \underline{0} & I \end{array}\right)\begin{pmatrix} \underline{x} \\ \underline{\mu} \\ \underline{\lambda} \\ \underline{s} \end{pmatrix} = \begin{pmatrix} \underline{c}^T \\ \underline{b} \end{pmatrix} \tag{4}$$

where $2D = H_f$ (the Hessian matrix of f)

$$\underline{\lambda}, \underline{\mu}, \underline{x}, \underline{s} \geq 0$$

2.2 Developed Technique: Inequality Constrained Quadratic Programming Problems via Residual Minimization Approach (ICQPP-RMA)

This developed approach handles an optimization problem whose objective function is quadratic and constraints are linear with inequality signs; thus, known as inequality constrained quadratic programming problems (ICQPP). Mathematically, an ICQPP is defined as:

$$\left.\begin{aligned} &Max.(or\ Min.)\ z = \underline{c}^T \underline{x} + \frac{1}{2}\underline{x}^T B\underline{x} \\ &Subject\ to: A\underline{x} \leq (or \geq)\underline{b} \\ &\qquad \underline{x} \geq \underline{0} \end{aligned}\right\} \tag{5}$$

where $\underline{c}, \underline{x}, \underline{b}, A\ and\ B = 2D$ *are as defined in* Equation (1)

2.2.1 Description of the Developed Technique in Stages

Stage 1: A matrix form representation of the objective function: The quadratic function should be expressed as:

$$z = \underline{c}^T \underline{x} + \frac{1}{2}\underline{x}^T B\underline{x}$$

Stage 2: Stationarity equation derivation: The gradient of z is taking with respect to $\underline{x}$ and equated to zero, that is;

$$B\underline{x} + \underline{c} = \underline{0} \qquad \text{(Condition for Stationarity)}$$

In addition, the candidate minimizer $\underline{x}^*$ is solved for the linear system which is unconstrained.

Stage 3: Unconstrained solution is checked for feasibility: Examine if $\underline{x}^*$ meets/satisfies the following;

(i) $A\underline{x}^* \leq \underline{b}$

(ii) $\underline{x}^* \geq \underline{0}$.

Thus, if (i) and (ii) are satisfied, $\underline{x}^*$ becomes the optimal solution, otherwise go to stage 4.

Stage 4: Infeasibility attracts constraint(s) activation: When $\underline{x}^*$ does not satisfy any constraint, compel the binding constraint(s) to stand as equations (that is, activate it). Additionally,

(i) Choose a constraint (or combination) more likely to be active at optimality,

(ii) Evaluate the active constraint(s) for one or more variables in terms of the others, by reducing the number of free variables

Stage 5: Problem dimension reduction: Put the expression (s) from the active constraint into the original objective function. This simplifies the problem to a lower-dimensional quadratic expression involving fewer variables.

Stage 6: Restate and evaluate the simplified version of the problem: This is done by:

(i) Building/constructing a simplified matrix representation:

$$z = \underline{c}_\tau^T \underline{y} + \frac{1}{2} \underline{y}^T B_\tau \underline{x}$$

where $\underline{y}$ is the simplified/reduced vector of free variables

(ii) Evaluate the simplified(reduced) condition for stationarity:

$$B_\tau \underline{y} + \underline{c}_\tau = \underline{0}$$

Stage 7: Rebuild complete set of variable: Put the optimal values of the free variables back to get $\underline{x}$, ensuring that every constraint is fulfilled.

$$A\underline{x} \leq \underline{b}, \ \underline{x} \geq \underline{0}$$

Stage 8: Compute the objective function: This is done by computing $z_{\min}$ ($z_{\max}$).

2.2.2 Wolfram Mathematica Scripts for the Three Problems via ICQPP-RMA

Example 1

```
ClearAll[x1, x2, B, c, x, grad, constraints, isFeasible, z1, z, solVec];
(* Step 1: Time and main logic *)
{execTime, result} = Timing[
 Module[{B, c, x, grad, unconSol, constraints, zVal, optX1, optX2,
   optZ, x1sub, constraintCheck, z1Expr, BVal},
   (* Step 2: Convert maximization to minimization form *)
   B = {{4, 2}, {2, 4}};
   c = {-4, -6}; (* Negative of linear part *)
   x = {x1, x2};
   (* Step 3: Solve Bx + c = 0 *)
   grad = B.x + c;
   unconSol = Solve[grad == {0, 0}, x];
   (* Step 4: Define constraints *)
   constraints = {
    x1 + 2 x2 <= 2,
    x1 >= 0,
    x2 >= 0
   };
   (* Step 5: Define feasibility checker *)
   isFeasible[sol_List] := Module[{vals},
    vals = x /. sol;
    And @@ (constraints /. Thread[x -> vals])
   ];
   Print["Checking unconstrained solution..."];
   If[Length[unconSol] > 0 && isFeasible[unconSol[[1]]],
    Print["✓ Unconstrained solution feasible."];
    solVec = x /. unconSol[[1]];
    Print["x1 = ", solVec[[1]], ", x2 = ", solVec[[2]]];
    (* Evaluate original max objective function directly *)
    z[x1_, x2_] := 4 x1 + 6 x2 - 2 x1^2 - 2 x1 x2 - 2 x2^2;
    zVal = z @@ solVec;
    Print["✓ Final Maximum z = ", zVal];
    (* Check constraint satisfaction *)
    constraintCheck =
     And[solVec[[1]] + 2 solVec[[2]] <= 2,
       solVec[[1]] >= 0, solVec[[2]] >= 0];
    Print["All constraints satisfied? ", constraintCheck],
    (* ✗ Enforce boundary constraint x1 + 2 x2 == 2 *)
    Print["✗ Unconstrained solution not feasible. Enforcing constraint x1 + 2x2 == 2"];
    x1sub = Solve[x1 + 2 x2 == 2, x1][[1]];
    z1Expr = Simplify[
     (4 x1 + 6 x2 - 2 x1^2 - 2 x1 x2 - 2 x2^2) /. x1sub
    ];
    z1[x2_] := z1Expr;
    optX2 =
     x2 /. Solve[
```

```
      D[z1[x2], x2] == 0 && x2 >= 0 && (x1 /. x1sub) >= 0, x2
     ][[1]];
    optX1 = x1 /. x1sub /. x2 -> optX2;
    (* Final z value *)
    z[x1_, x2_] := 4 x1 + 6 x2 - 2 x1^2 - 2 x1 x2 - 2 x2^2;
    optZ = z[optX1, optX2];
    Print["✅ Constrained optimal solution found."];
    Print["x1 = ", optX1];
    Print["x2 = ", optX2];
    Print["✅ Final Maximum z = ", optZ];
    constraintCheck =
     And[optX1 + 2 optX2 <= 2, optX1 >= 0, optX2 >= 0];
    Print["All constraints satisfied? ", constraintCheck];
   ]
  ]
];
(* Step 6: Show CPU time *)
Print["\n⏱ CPU Execution Time: ", NumberForm[execTime, {4, 6}], " seconds"];
```

Example 2

```
ClearAll[x1, x2, B, c, x, grad, constraints, isFeasible, z1, z];
(* Step 0: Use both Timing and AbsoluteTiming *)
timingResults = Timing[
 absoluteResults = AbsoluteTiming[
   Module[
    {B, c, x, grad, unconSol, constraints, solVec, optX2, optX1,
     optZ, x1sub, constraintCheck},
    (* Step 1: Convert maximization to minimization *)
    B = {{-4, -2}, {-2, -4}};
    c = {-12, -21};
    x = {x1, x2};
    (* Step 2: Solve Bx + c = 0 *)
    grad = B.x + c;
    unconSol = Solve[grad == {0, 0}, x];
     (* Step 3: Define constraints *)
    constraints = {
     x2 <= 8,
     x1 + x2 <= 10,
     x1 >= 0,
     x2 >= 0
    };
    (* Step 4: Feasibility checker *)
    isFeasible[sol_List] := Module[{vals},
     vals = x /. sol;
     And @@ (constraints /. Thread[x -> vals])
    ];
    Print["Checking unconstrained solution..."];
    If[Length[unconSol] > 0 && isFeasible[unconSol[[1]]],
```

```
      Print["✅ Unconstrained solution feasible."];
      solVec = x /. unconSol[[1]];
      z[x1_, x2_] := 12 x1 + 21 x2 + 2 x1 x2 - 2 x1^2 - 2 x2^2;
      optZ = z @@ solVec;
      Print["x1 = ", solVec[[1]], ", x2 = ", solVec[[2]]];
      Print["✅ Final Maximum z = ", optZ];
      constraintCheck =
       And[solVec[[2]] <= 8,
         solVec[[1]] + solVec[[2]] <= 10,
         solVec[[1]] >= 0, solVec[[2]] >= 0];
      Print["All constraints satisfied? ", constraintCheck],
      (* Else: Enforce boundary x1 + x2 == 10 *)
      Print["❌ Unconstrained solution not feasible. Enforcing constraint x1 + x2 == 10"];
      x1sub = Solve[x1 + x2 == 10, x1][[1]];
      z1[x2_] := (12 x1 + 21 x2 + 2 x1 x2 - 2 x1^2 - 2 x2^2) /. x1sub;
      optX2 =
       x2 /. Solve[
         D[z1[x2], x2] == 0 && x2 >= 0 && x2 <= 8 &&
          (x1 /. x1sub) >= 0,
         x2
        ][[1]];
      optX1 = x1 /. x1sub /. x2 -> optX2;
      z[x1_, x2_] := 12 x1 + 21 x2 + 2 x1 x2 - 2 x1^2 - 2 x2^2;
      optZ = z[optX1, optX2];
      Print["✅ Constrained optimal solution found."];
      Print["x1 = ", optX1];
      Print["x2 = ", optX2];
      Print["✅ Final Maximum z = ", optZ];
      constraintCheck =
       And[optX2 <= 8, optX1 + optX2 <= 10, optX1 >= 0, optX2 >= 0];
      Print["All constraints satisfied? ", constraintCheck];
     ]
    ]
   ];
];
(* Step 5: Print both times *)
Print["⏱ AbsoluteTiming = ",
  NumberForm[absoluteResults[[1]], {4, 6}], " seconds"];
Print["⚙ CPU Timing = ",
  NumberForm[timingResults[[1]], {4, 6}], " seconds"];
```

Example 3

```
ClearAll[x1, x2, x3, B, c, x, unconSol, grad, constraints, isFeasible, z, solVec];
(* Step 1: Define B and c *)
B = {{4, 2, 0}, {2, 4, 2}, {0, 2, 4}};
c = {0, -3, -5};
x = {x1, x2, x3};
(* Step 2: Use Timing to measure execution time *)
{execTime, dummy} =
```

```
Timing[
 Module[{},
  (* Step 3: Solve the unconstrained stationarity condition *)
  grad = B.x + c;
  unconSol = Solve[grad == {0, 0, 0}, x];
  (* Step 4: Define constraints *)
  constraints = {
   x1 + x2 + x3 >= 0,
   3 x1 + 2 x2 + x3 <= 6,
   x1 >= 0,
   x2 >= 0,
   x3 >= 0
  };
  (* Step 5: Feasibility checker *)
  isFeasible[sol_List] := Module[{vals},
   vals = x /. sol;
   And @@ (constraints /. Thread[x -> vals])
  ];
  (* Step 6: Try unconstrained solution first *)
  If[Length[unconSol] > 0 && isFeasible[unconSol[[1]]],
   Print["✅ Feasible unconstrained solution found."];
   solVec = x /. unconSol[[1]],
   (* Step 7: Try x1 = 0 (boundary case) *)
   Print["❌ Unconstrained solution not feasible. Activating constraint x1 = 0"];
   x1val = 0;
   Br = B[[2 ;;, 2 ;;]];
   cr = c[[2 ;;]] + B[[2 ;;, 1]]*x1val;
   gradRed = Br.{x2, x3} + cr;
   redSol = Solve[gradRed == {0, 0}, {x2, x3}];
   If[Length[redSol] > 0,
    fullSol = Join[{x1 -> x1val}, redSol[[1]]];
    If[isFeasible[fullSol],
     Print["✅ Feasible boundary solution found."];
     solVec = x /. fullSol,
     Print["❌ Boundary solution still not feasible."]; Abort[]
    ],
    Print["❌ Could not solve reduced system."]; Abort[]
   ]
  ];
  (* Step 8: Define and evaluate objective function *)
  z[x1_, x2_, x3_] := (1/2) {x1, x2, x3}.B.{x1, x2, x3} + c.{x1, x2, x3};
  zVal = z @@ solVec;
  (* Step 9: Check constraints *)
  constraintSat = And @@ (constraints /. Thread[x -> solVec]);
  (* Step 10: Display results *)
  Print["\n--- Solution Metrics ---"];
  Print["Solution: x1 = ", solVec[[1]], ", x2 = ", solVec[[2]], ", x3 = ", solVec[[3]]];
  Print["Minimum z = ", zVal];
  Print["All constraints satisfied? ", constraintSat];
 ]
```

```
 ];
(* Step 11: Print Timing result *)
Print["\n⏱ CPU Time (Timing) = ", NumberForm[execTime, {4, 6}], " seconds"];
```

3 Results

3.1 Employing the Existing Dantzig-Wolfe Algorithm to Solve Numerical Examples on ICQPP

Example 1: Solve the following ICQPP [11]

$$\begin{aligned} &Max.\ z = 4x_1 + 6x_2 - 2x_1^2 - 2x_1x_2 - 2x_2^2 \\ &Subject\ to: x_1 + 2x_2 \le 2 \\ &\qquad x_1, x_2 \ge 0 \end{aligned}$$

Solution:

$$\underline{c} = \begin{pmatrix} 4 & 6 \end{pmatrix} \qquad D = \begin{pmatrix} -2 & -1 \\ -1 & -2 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 2 \end{pmatrix} \qquad \underline{b} = (2)$$

Applying Equation (4) to get:

$$\begin{pmatrix} 4 & 2 & 1 & -1 & 0 & 0 \\ 2 & 4 & 2 & 0 & -1 & 0 \\ 1 & 2 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \mu \\ \lambda_1 \\ \lambda_2 \\ s \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 2 \end{pmatrix}$$

$$\left.\begin{aligned} 4x_1 + 2x_2 + \mu - \lambda_1 &= 4 \\ \Rightarrow 2x_1 + 4x_2 + 2\mu - \lambda_2 &= 6 \\ x_1 + 2x_2 + S &= 2 \end{aligned}\right\} \tag{6}$$

Adding artificial variables (R_1, R_2) to the first two equations of Equation (6) in order to get the initial basic feasible solution produces equation (7).

$$\left.\begin{aligned} 4x_1 + 2x_2 + \mu - \lambda_1 + R_1 &= 4 \\ 2x_1 + 4x_2 + 2\mu - \lambda_2 + R_2 &= 6 \\ x_1 + 2x_2 + S &= 2 \end{aligned}\right\} \tag{7}$$

Employing phase-I of the two-phase method of simplex method to get Equation (8)

$$\left.\begin{aligned} &Min.\ r_0 = R_1 + R_2 \\ &Subject\ to: 4x_1 + 2x_2 + \mu - \lambda_1 + R_1 = 4 \\ &\qquad 2x_1 + 4x_2 + 2\mu - \lambda_2 + R_2 = 6 \\ &\qquad x_1 + 2x_2 + S = 2 \\ &x_1, x_2, \mu, \lambda_1, \lambda_2, R_1, R_2, S \geq 0 \end{aligned}\right\} \quad (8)$$

Table 1: Phase-1 of Two Phase Simplex Method for Example 1

		N					B			$\underline{b}$
Basic	r_0	x_1	x_2	μ	λ_1	λ_2	R_1	R_2	S	RHS
r_0	(1)	0	0	0	0	-1	-1	-1	0	0
R_1	0	4	2	1	-1	0	1	0	0	4
R_2	0	2	4	2	0	-1	0	1	0	6
S	0	1	2	0	0	0	0	0	1	2
r_0	(1)	6	6	3	-1	-1	0	0	0	10
R_1	0	4	2	1	-1	0	1	0	0	4
R_2	0	2	4	2	0	-1	0	1	0	6
S	0	1	2	0	0	0	0	0	1	2
r_0	0	0	3	3/2	1/2	-1	-3/2	0	0	4
x_1	0	1	½	¼	-1/4	0	¼	0	0	1
R_2	0	0	3	3/2	1/2	-1	-1/2	1	0	4
S	0	0	3/2	-1/4	1/4	0	-1/4	0	1	1
r_0	0	0	0	2	0	-1	-1	0	-2	2
x_1	0	1	0	1/3	-1/3	0	1/3	0	-1/3	2/3
R_2	0	0	0	2	0	-1	0	1	-2	2
x_2	0	0	1	-1/6	1/6	0	-1/6	0	2/3	2/3
r_0	0	0	0	0	0	0	-1	-1	0	0
x_1	0	1	0	0	-1/3	1/6	1/3	-1/6	0	1/3
μ	0	0	0	1	0	-1/2	0	1/2	-1	1
x_2	0	0	1	0	1/6	-1/12	-1/6	1/12	1/2	5/6

The final tableau of Table 1 is optimal and gives the required solution

Therefore $x_1 = 1/3, x_2 = 5/6, \mu = 1, \lambda_1 = \lambda_2 = 0$ and $Z = 25/6$

Example 2: Solve the following ICQPP [12]

$$\begin{aligned} &Max.\ z = 12x_1 + 21x_2 - 2x_1^2 + 2x_1x_2 - 2x_2^2 \\ &Subject\ to: x_2 \le 8 \\ &\qquad x_1 + x_2 \le 10 \\ &\qquad x_1, x_2 \ge 0 \end{aligned}$$

Solution:

$$\underline{c} = \begin{pmatrix} 12 & 21 \end{pmatrix} \quad D = \begin{pmatrix} -2 & 1 \\ 1 & -2 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \qquad \underline{b} = \begin{pmatrix} 8 \\ 10 \end{pmatrix}$$

Applying Equation (4) to get:

$$\begin{pmatrix} 4 & -2 & 0 & 1 & -1 & 0 & 0 & 0 \\ -2 & 4 & 1 & 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \mu_1 \\ \mu_2 \\ \lambda_1 \\ \lambda_2 \\ s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 12 \\ 21 \\ 8 \\ 10 \end{pmatrix}$$

$$\Rightarrow \left.\begin{aligned} 4x_1 - 2x_2 + \mu_2 - \lambda_1 &= 12 \\ -2x_1 + 4x_2 + \mu_1 + \mu_2 - \lambda_2 &= 21 \\ x_2 + S_1 &= 8 \\ x_1 + x_2 + S_2 &= 10 \end{aligned}\right\} \tag{9}$$

Adding artificial variables (R_1, R_2) to the first two equations of Equation (9) in order to get the initial basic feasible solution produces equation (10).

$$\left.\begin{aligned} 4x_1 - 2x_2 + \mu_2 - \lambda_1 + R_1 &= 12 \\ -2x_1 + 4x_2 + \mu_1 + \mu_2 - \lambda_2 + R_2 &= 21 \\ x_2 + S_1 &= 8 \\ x_1 + x_2 + S_2 &= 10 \end{aligned}\right\} \tag{10}$$

Employing phase-I of the two-phase method of simplex method to get Equation (11)

$$\left.\begin{aligned} &Min.\ r_0 = R_1 + R_2 \\ &Subject\ to: 4x_1 - 2x_2 + \mu_2 - \lambda_1 + R_1 = 12 \\ &\quad -2x_1 + 4x_2 + \mu_1 + \mu_2 - \lambda_2 + R_2 = 21 \\ &\quad x_2 + S_1 = 8 \\ &\quad x_1 + x_2 + S_2 = 10 \\ &x_1, x_2, \mu_1, \mu_2, \lambda_1, \lambda_2, R_1, R_2, S_1, S_2 \geq 0 \end{aligned}\right\} \quad (11)$$

Table 2: Phase-1 of Two Phase Simplex Method for Example 2

		N						B				$\underline{b}$
Basic	r_0	x_1	x_2	μ_1	μ_2	λ_1	λ_2	R_1	R_2	S_1	S_2	RHS
r_0	(1)	0	0	0	0	0	0	-1	-1	0	0	0
R_1	0	4	-2	0	1	-1	0	1	0	0	0	12
R_2	0	-2	4	1	1	0	-1	0	1	0	0	21
S_1	0	0	1	0	0	0	0	0	0	1	0	8
S_2	0	1	1	0	0	0	0	0	0	0	1	10
r_0	(1)	2	2	1	2	-1	-1	0	0	0	0	33
R_1	0	4	-2	0	1	-1	0	1	0	0	0	12
R_2	0	-2	4	1	1	0	-1	0	1	0	0	21
S_1	0	0	1	0	0	0	0	0	0	1	0	8
S_2	0	1	1	0	0	0	0	0	0	0	1	10
r_0	0	0	3	1	3/2	-1/2	-1	-1/2	0	0	0	27
x_1	0	1	-1/2	0	¼	-1/4	0	1/4	0	0	0	3
R_2	0	0	3	1	3/2	-1/2	-1	1/2	1	0	0	27
S_1	0	0	1	0	0	0	0	0	0	1	0	8
S_2	0	0	3/2	0	-1/4	1/4	0	-1/4	0	0	1	7
r_0	0	0	0	1	2	-1	-1	0	0	0	-2	13
x_1	0	1	0	0	1/6	-1/6	0	1/6	0	0	1/3	16/3
R_2	0	0	0	1	2	-1	-1	1	1	0	-2	13
S_1	0	0	0	0	1/6	-1/6	0	1/6	0	1	-2/3	10/3
x_2	0	0	1	0	-1/6	1/6	0	-1/6	0	0	2/3	14/3
r_0	0	0	0	0	0	0	0	-1	-1	0	0	0
x_1	0	1	0	-1/12	0	-1/12	1/12	1/12	-1/12	0	½	17/4
μ_2	0	0	0	½	1	-1/2	-1/2	1/2	½	0	-1	13/2
S_1	0	0	0	-1/12	0	-1/12	1/12	1/12	-1/12	1	-1/2	9/4
x_2	0	0	1	1/12	0	1/12	-1/12	-1/12	1/12	0	½	23/4

The final tableau of Table 2 is optimal and gives the required solution
Therefore $x_1 = 17/4,\ x_2 = 23/4,\ \mu_2 = 13/2,\ S_1 = 9/4,\ S_2 = \mu_1 = \lambda_1 = \lambda_2 = 0$ and $Z = 947/8$

Example 3: Solve the following ICQPP [11]

$$\begin{aligned} &Min.\ z = 2x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3 + x_1 - 3x_2 - 5x_3 \\ &Subject\ to: x_1 + x_2 + x_3 \geq 0 \\ &\qquad 3x_1 + 2x_2 + x_3 \leq 6 \\ &\qquad x_1, x_2, x_3 \geq 0 \end{aligned}$$

Solution:

Converting the problem to a maximization type and putting it into symmetric form produces

$$\begin{aligned} &Max.\ z = -2x_1^2 - 2x_2^2 - 2x_3^2 - 2x_1x_2 - 2x_2x_3 - x_1 + 3x_2 + 5x_3 \\ &Subject\ to: -x_1 - x_2 - x_3 \leq 0 \\ &\qquad 3x_1 + 2x_2 + x_3 \leq 6 \\ &\qquad x_1, x_2, x_3 \geq 0 \end{aligned}$$

$$\underline{c} = \begin{pmatrix} -1 & 3 & 5 \end{pmatrix} \qquad D = \begin{pmatrix} -2 & -1 & 0 \\ -1 & -2 & -1 \\ 0 & -1 & -2 \end{pmatrix} \qquad A = \begin{pmatrix} -1 & -1 & -1 \\ 3 & 2 & 1 \end{pmatrix} \qquad \underline{b} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}$$

Applying Equation (4) to get:

$$\begin{pmatrix} 4 & 2 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ 2 & 4 & 2 & -1 & 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 2 & 4 & -1 & 1 & 0 & 0 & -1 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \mu_1 \\ \mu_2 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \\ 5 \\ 0 \\ 6 \end{pmatrix}$$

$$\left.\begin{aligned} 4x_1 + 2x_2 - \mu_1 + 3\mu_2 - \lambda_1 &= -1 \\ 2x_1 + 4x_2 + 2x_3 - \mu_1 + 2\mu_2 - \lambda_2 &= 3 \\ \Rightarrow 2x_2 + 4x_3 - \mu_1 + \mu_2 - \lambda_3 &= 5 \\ -x_1 - x_2 - x_3 + S_1 &= 0 \\ 3x_1 + 2x_2 + x_3 + S_2 &= 6 \end{aligned}\right\} \quad (12)$$

Multiply through the first equation in Equation (12) by -1 and add artificial variables (R_1, R_2, R_3) to the first three equations of Equation (12) in order to get the initial basic feasible solution to get Equation (13).

$$\left.\begin{aligned} &Min.\ r_0 = R_1 + R_2 + R_3 \\ &Subject\ to: -4x_1 - 2x_2 + \mu_1 - 3\mu_2 + \lambda_1 + R_1 = 1 \\ &2x_1 + 4x_2 + 2x_3 - \mu_1 + 2\mu_2 - \lambda_2 + R_2 = 3 \\ &2x_2 + 4x_3 - \mu_1 + \mu_2 - \lambda_3 + R_3 = 5 \\ &-x_1 - x_2 - x_3 + S_1 = 0 \\ &3x_1 + 2x_2 + x_3 + S_2 = 6 \\ &x_1, x_2, x_3, \mu_1, \mu_2, \lambda_1, \lambda_2, \lambda_3, R_1, R_2, R_3, S_1, S_2 \geq 0 \end{aligned}\right\} \quad (13)$$

Table 3: Phase-1 of Two Phase Simplex Method for Example 3

		N								B					$\underline{b}$
Basic	r_0	x_1	x_2	x_3	μ_1	μ_2	λ_1	λ_2	λ_3	R_1	R_2	R_3	S_1	S_2	RHS
r_0	(1)	0	0	0	0	0	0	0	0	-1	-1	-1	0	0	0
R_1	0	-4	-2	0	1	-3	0	0	0	1	0	0	0	0	1
R_2	0	-2	4	2	-1	2	0	-1	0	0	1	0	0	0	3
R_3	0	0	2	4	-1	1	0	0	-1	0	0	1	0	0	5
S_1	0	-1	-1	-1	0	0	0	0	0	0	0	0	1	0	0
S_2	0	3	2	1	0	0	0	0	0	0	0	0	0	1	6
r_0	(1)	-2	4	6	-1	0	1	-1	-1	0	0	0	0	0	9
R_1	0	-4	-2	0	1	-3	1	0	0	1	0	0	0	0	1
R_2	0	2	4	2	-1	2	0	-1	0	0	1	0	0	0	3
R_3	0	0	2	4	-1	1	0	0	-1	0	0	1	0	0	5
S_1	0	-1	-1	-1	0	0	0	0	0	0	0	0	1	0	0
S_2	0	3	2	1	0	0	0	0	0	0	0	0	0	1	6
r_0	0	-2	1	0	1/2	-3/2	1	-1	1/2	1	0	-3/2	0	0	3/2
R_1	0	-4	-2	0	1	-3	1	0	0	0	0	0	0	0	1
R_2	0	2	3	0	-1/2	3/2	0	-1	1/2	0	1	-1/2	0	0	½
x_3	0	0	1/2	1	-1/4	¼	0	0	-1/4	0	0	1/4	0	0	5/4
S_1	0	-1	-1/2	0	-1/4	¼	0	0	-1/4	0	0	1/4	1	0	5/4
S_2	0	3	3/2	0	1/4	-1/4	0	0	1/4	0	0	-1/4	0	1	19/4
r_0	0	-8/3	0	0	2/3	-2	1	-2/3	1/3	0	-1/3	-4/3	0	0	4/3
R_1	0	-8/3	0	0	2/3	-2	1	-2/3	1/3	1	2/3	-1/3	0	0	4/3
x_2	0	2/3	1	0	-1/6	½	0	-1/3	1/6	0	1/3	-1/6	0	0	1/6
x_3	0	-1/3	0	1	-1/6	0	0	1/6	-1/3	0	-1/6	1/3	0	0	7/6
S_1	0	-2/3	0	0	-1/3	½	0	-1/6	-1/6	0	1/6	1/6	1	0	4/3
S_2	0	2	0	0	1/2	-1	0	1/2	0	0	-1/2	0	0	1	9/2
r_0	0	0	0	0	0	0	0	0	0	-1	-1	-1	0	0	0
λ_1	0	-8/3	0	0	2/3	-2	1	-2/3	1/3	1	2/3	-1/3	0	0	4/3
x_2	0	2/3	1	0	-1/6	½	0	-1/3	1/6	0	1/3	-1/6	0	0	1/6

x_3	0	-1/3	0	1	-1/6	0	0	1/6	-1/3	0	-1/6	1/3	0	0	7/6
S_1	0	-2/3	0	0	-1/3	½	0	-1/6	-1/6	0	1/6	1/6	1	0	4/3
S_2	0	2	0	0	1/2	-1	0	1/2	0	0	-1/2	0	0	1	9/2

The current tableau of Table 3 is optimal. Hence the solution is
$x_1 = 0, x_2 = 1/6, x_3 = 7/6, S_1 = 4/3, S_2 = 9/2, \lambda_1 = 4/3, \lambda_2 = \lambda_3 = \mu_1 = \mu_2 = 0$ and $Z = -19/6$

3.2 Employing the Developed ICQPP-RMA to Illustrate Numerical Examples

Example 1

Using example 1, the objective function which is in maximization is first converted to minimization, that is:

$$Min.\ z = -4x_1 - 6x_2 + 2x_1^2 + 2x_1x_2 + 2x_2^2$$

The objective function is now in quadratic form of $z = \underline{c}^T \underline{x} + \frac{1}{2}\underline{x}^T B\underline{x}$

Where $\underline{c}^T = \begin{pmatrix} -4 & -6 \end{pmatrix}$ $B = 2D = 2\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}$

Let $B\underline{x} = -\underline{c} + \varepsilon$ (ε is the residual for the regression), considering the problem as a constrained regression minimization. Then, the minimization gives

$$\underset{\underline{x}}{Min} \left\| B\underline{x} + \underline{c} \right\|^2 \quad Subject\ to: x_1 + 2x_2 \le 2; \quad x_1, x_2 \ge 0$$

The system is solved using least squares as;

$$B\underline{x} = -\underline{c} \Rightarrow \begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \end{pmatrix}$$

$$4x_1 + 2x_2 = 4 \quad (14)$$

$$2x_1 + 4x_2 = 6 \quad (15)$$

Equation (15) × 2: $4x_2 + 8x_2 = 12$ (16)

Equation (16) – Equation (14): $6x_2 = 8 \Rightarrow x_2 = 4/3$

Put $x_2 = 4/3$ into Equation (15) to get $2x_1 + 4(4/3) = 6 \Rightarrow 2x_1 + 16/3 = 6 \Rightarrow 2x_1 = 6 - 16/3$

$2x_1 = 2/3 \Rightarrow x_1 = 1/3$

Thus, the solution of the unconstrained least squares is $x_1 = 1/3, x_2 = 4/3$

Checking Feasibility Constraint:

$x_1 + 2x_2 = 1/3 + 2(4/3) = 3 > 2 \Rightarrow$ *i*nfeasible

Feasibility Enforcement

Putting the constraint directly to reduce problem size, that is

$$x_1 = 2 - 2x_2$$

$$z(x_2) = 4x_1 + 6x_2 - 2x_1^2 - 2x_1x_2 - 2x_2^2 \Rightarrow z(x_2) = 4(2 - 2x_2) + 6x_2 - 2(2 - 2x_2)^2 - 2x_2(2 - 2x_2) - 2x_2^2$$

$$z(x_2) = 8 - 8x_2 + 6x_2 - 2(4 - 8x_2 + 4x_2^2) - 4x_2 + 4x_2^2 - 2x_2^2 = 10x_2 - 6x_2^2$$

Optimize

Maximize: $z = 10x_2 - 6x_2^2$

$$\frac{dz(x_2)}{dx_2} = 10 - 12x_2 = 0 \Rightarrow x_2 = \frac{5}{6}$$

$$\therefore x_1 = 2 - 2(5/6) = 1/3$$

Therefore the final solution becomes $x_1 = 1/3, x_2 = 5/6$ *and* $z_{\max} = 25/6$

Example 2

Using example 2, the objective function which is in maximization is first converted to minimization, that is:

$$Min.\, z = -12x_1 - 21x_2 - 2x_1x_2 + 2x_1^2 + 2x_2^2$$

The objective function is now in quadratic form of $z = \underline{c}^T \underline{x} + \frac{1}{2}\underline{x}^T B\underline{x}$

Where $\underline{c}^T = \begin{pmatrix} -12 & -21 \end{pmatrix}$ $\quad B = 2D = 2\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} = \begin{pmatrix} 4 & -2 \\ -2 & 4 \end{pmatrix}$

Let $B\underline{x} = -\underline{c} + \varepsilon$ (ε is the residual for the regression), considering the problem as a constrained regression minimization. Then, the minimization gives

$$\underset{\underline{x}}{Min} \left\| B\underline{x} + \underline{c} \right\|^2 \quad Subject\ to: x_2 \le 8, \quad x_1 + x_2 \le 10; \quad x_1, x_2 \ge 0$$

The system is solved using least squares as;

$$B\underline{x} = -\underline{c} \Rightarrow \begin{pmatrix} 4 & -2 \\ -2 & 4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -12 \\ -21 \end{pmatrix}$$

$$4x_1 - 2x_2 = -12 \tag{17}$$

$$-2x_1 + 4x_2 = -21 \tag{18}$$

Equation (17) × 2: $18x_2 - 4x_2 = -24$ (19)

Equation (18) + Equation (19): $6x_1 = -45 \Rightarrow x_1 = -15/2$

Put $x_1 = -15/2$ into Equation (18) to get

$$-2(-15/2) + 4x_2 = -21 \Rightarrow 4x_2 = -21 - 15 = -36 \Rightarrow x_2 = -9$$

Thus, the solution of the unconstrained least squares is $x_1 = -15/2, x_2 = -9$

Checking Feasibility Constraint:

The both variables are negative, which implies not feasible

Feasibility Enforcement

Putting the active constraint directly to reduce problem size, that is

$$x_1 = 10 - x_2$$

$$z = 12x_1 + 12x_2 + 2x_1x_2 - 2x_1^2 - 2x_2^2 \Rightarrow z(x_2) = 12(10 - x_2) + 21x_2 + 2x_2(10 - x_2) - 2(10 - x_2)^2 - 2x_2^2$$

$$z(x_2) = 120 - 12x_2 + 21x_2 + 20x_2 - 2x_2^2 - 2(100 - 20x_2 + x_2^2) - x_2^2 = -80 + 69x_2 - 6x_2^2$$

Optimize

Maximize: $z = -80 + 69x_2 - 6x_2^2$

$$\frac{dz(x_2)}{dx_2} = 69 - 12x_2 = 0 \Rightarrow x_2 = \frac{23}{4}$$

$$\therefore x_1 = 10 - 23/4 = 17/4$$

Check Constraints

$x_2 = 23/4 \le 8$ Satisfied

$x_1 + x_2 = 17/4 + 23/4 = 10 \le 10$ Satisfied

Non-negativity test is satisfied as well

Therefore the final solution becomes $x_1 = 17/4, x_2 = 23/4$ *and* $z_{\max} = 947/8$

Example 3

Using example 3, the objective function which is already in minimization form, that is:

$Min.\ z = 2x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3 - 3x_2 - 5x_3$

The objective function is now in quadratic form of $z = \underline{c}^T \underline{x} + \frac{1}{2}\underline{x}^T B\underline{x}$

Where $\underline{c}^T = \begin{pmatrix} 0 & -3 & -5 \end{pmatrix}$ $\quad B = 2D = 2\begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 2 & 0 \\ 2 & 4 & 2 \\ 0 & 2 & 4 \end{pmatrix}$

Let $B\underline{x} = -\underline{c} + \varepsilon$ (ε is the residual for the regression), considering the problem as a constrained regression minimization. Then, the minimization gives

$$\underset{\underline{x}}{Min} \left\| B\underline{x} + \underline{c} \right\|^2 \quad Subject\ to: x_1 + x_2 + x_3 \ge 0,\ 3x_1 + 2x_2 + x_3 \le 6,\ x_1, x_2, x_3 \ge 0$$

The unconstrained system is solved using least squares as;

$$B\underline{x} = -\underline{c} \Rightarrow \begin{pmatrix} 4 & 2 & 0 \\ 2 & 4 & 2 \\ 0 & 2 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \\ 5 \end{pmatrix}$$

$$4x_1 + 2x_2 = 0 \tag{20}$$

$$2x_1 + 4x_2 + 2x_3 = 3 \tag{21}$$

$$2x_2 + 4x_3 = 5 \tag{22}$$

From Equation (1), $x_1 = -x_2 / 2$ (23)

Put Equation (23) into Equation (21) to get

$$2(-x_2 / 2) + 4x_2 + 2x_3 = 3 \Rightarrow -x_2 + 4x_2 + 2x_3 = 3$$

$$\therefore 3x_2 + 2x_3 = 3 \tag{24}$$

Equation (24) ×2: $6x_2 + 4x_3 = 6$ (25)

Equation (25) − Equation (22): $4x_2 = 1 \Rightarrow x_2 = 1/4$

Put $x_2 = 1/4$ into Equation (24) to get $3(1/4) + 2x_3 = 3 \Rightarrow 2x_3 = 3 - 3/4 \Rightarrow x_3 = 9/8$

Put $x_2 = 1/4$ into Equation (23) to get $x_1 = -1/4 \times 1/2 = -1/8$

Thus, the solution of the unconstrained least squares is $x_1 = -1/8, x_2 = 1/4, x_3 = 9/8$

Checking Feasibility Constraint:

Since $x_1 = -1/8$ is negative, this point violates the non-negativity constraint, hence $x_1 \geq 0$ is not feasible.

Feasibility Enforcement

Since the unconstrained optimum lies outside the feasible region, a constraint is then forced to be active, for instance, assume $x_1 = 0$, to reduce the problem to x_2, x_3

Then, set up reduced 2-variable problem and minimize

$$z = 2x_2^2 + 2x_3^2 + 2x_2x_3 - 3x_2 - 5x_3 \ \textit{Subject to}: x_2 + x_3 \geq 0, \ 2x_2 + x_3 \leq 6, x_2, x_3 \geq 0$$

Where $\underline{x}^T = \begin{pmatrix} x_1 & x_2 \end{pmatrix}$ $\underline{c}^T = \begin{pmatrix} -3 & -5 \end{pmatrix}$ $B = 2D = 2\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}$

The system is solved using least squares as;

$$B\underline{x} = -\underline{c} \Rightarrow \begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$$

$$4x_2 + 2x_3 = 3 \quad (26)$$

$$2x_2 + 4x_3 = 5 \quad (27)$$

Equation (26) × 2: $8x_2 + 4x_3 = 6$ (28)

Equation (28) − Equation (27): $6x_2 = 1 \Rightarrow x_2 = 1/6$

Put $x_2 = 1/6$ into Equation (27) to get $2(1/6) + 4x_3 = 5 \Rightarrow 4x_3 = 5 - 1/3 = 14/3 \Rightarrow x_3 = 7/6$

Thus, the solution of the unconstrained least squares is $x_1 = 0,\ x_2 = 1/6,\ x_3 = 7/6$

Check Constraints:

$x_1 + x_2 + x_3 = 0 + 1/6 + 7/6 = 4/3 > 0$ Satisfied

$3x_1 + 2x_2 + x_3 = 3(0) + 2(1/6) + 7/6 = 3/2 < 6$ Satisfied

Non-negativity test is satisfied as well ($x_1, x_2, x_3 \geq 0$)

Therefore the final solution becomes $x_1 = 0, x_2 = 1/6,\ x_3 = 7/6$ *and* $z_{\min} = -19/6$

Table 4: Performance Summary Results of the Problem Sets via Wolfram Mathematica

Problem set	Mathematica functions/methods	CPU execution time (s)	Optimal solution	Objective value	Constraint satisfaction?
Example 1	Maximize/Minimize	0.031250	$x_1 = 1/3, x_2 = 5/6$	$z_{\max} = 25/6$	Satisfied
	NMaximize/NMinimize	0.151275	$x_1 = 1/3, x_2 = 5/6$	$z_{\max} = 25/6$	Satisfied
	FindMaximum/FindMinimum	0.015625	$x_1 = 1/3, x_2 = 5/6$	$z_{\max} = 25/6$	Satisfied
	Dantzig-Wolfe Algorithm	0.002409	$x_1 = 1/3, x_2 = 5/6$	$z_{\max} = 25/6$	Satisfied
	ICQPP-RMA	0.000000	$x_1 = 1/3, x_2 = 5/6$	$z_{\max} = 25/6$	Satisfied
Example 2	Maximize/Minimize	0.062500	$x_1 = 17/4, x_2 = 23/4$	$z_{\max} = 947/8$	Satisfied
	NMaximize/NMinimize	0.187500	$x_1 = 17/4, x_2 = 23/4$	$z_{\max} = 947/8$	Satisfied
	FindMaximum/FindMinimum	0.031250	$x_1 = 17/4, x_2 = 23/4$	$z_{\max} = 947/8$	Satisfied
	Dantzig-Wolfe Algorithm	0.002953	$x_1 = 17/4, x_2 = 23/4$	$z_{\max} = 947/8$	Satisfied
	ICQPP-RMA	0.000000	$x_1 = 17/4, x_2 = 23/4$	$z_{\max} = 947/8$	Satisfied
Example 3	Maximize/Minimize	0.093750	$x_1 = 0, x_2 = 1/6,\ x_3 = 7/6$	$z_{\min} = -19/6$	Satisfied
	NMaximize/NMinimize	0.203125	$x_1 = 0, x_2 = 1/6,\ x_3 = 7/6$	$z_{\min} = -19/6$	Satisfied
	FindMaximum/FindMinimum	0.046875	$x_1 = 0, x_2 = 1/6,\ x_3 = 7/6$	$z_{\min} = -19/6$	Satisfied
	Dantzig-Wolfe Algorithm	0.003416	$x_1 = 0, x_2 = 1/6,\ x_3 = 7/6$	$z_{\min} = -19/6$	Satisfied
	ICQPP-RMA	0.000000	$x_1 = 0, x_2 = 1/6,\ x_3 = 7/6$	$z_{\min} = -19/6$	Satisfied

Table 4 shows a comparative performance analysis of three computational methods used in Wolfram Mathematica to solve three sets of inequality-constrained quadratic programming problems, as well as an existing method and developed method. The methods include the standard symbolic approach (Maximize/Minimize), numerical global optimization (NMaximize/NMinimize), local optimization techniques (FindMaximum/FindMinimum), the Dantzig-Wolfe decomposition algorithm, and a novel technique termed ICQPP-RMA (Inequality Constrained Quadratic Programming via Residual Minimization Approach). Across all three problem sets, the five methods consistently produced the same optimal solutions and objective function values, confirming their mathematical correctness and reliability. Additionally, in each case, the computed solutions satisfied the given constraints, indicating robust constraint handling across methods. However, the most notable differences are found in the CPU execution times. The symbolic Maximize/Minimize method exhibited moderate computational times, ranging between 0.03125 and 0.09375 seconds, reflecting the overhead typically associated with symbolic processing. The NMaximize/NMinimize method, though accurate, was the slowest overall, with execution times as high as 0.203125 seconds, due to its reliance on global numerical optimization heuristics. In contrast, FindMaximum/FindMinimum, which applies local optimization strategies, performed more efficiently, with CPU times significantly lower than those of NMinimize, albeit dependent on good starting values.

The Dantzig-Wolfe algorithm, implemented likely as a structured decomposition method, demonstrated exceptional efficiency with execution times between 0.002409 and 0.003416 seconds. Its performance illustrates the advantage of leveraging problem structure in optimization. However, the most outstanding performance was recorded by the proposed ICQPP-RMA, which achieved the same optimal results with a reported execution time of 0.000000 seconds across all problem instances. This suggests either negligible computational load due to a highly efficient formulation or symbolic pre-processing that bypasses the need for iterative numerical procedures.

4 Conclusion

The study concludes that the proposed ICQPP-RMA method is not only accurate but also the most computationally efficient approach among the methods evaluated for solving inequality-constrained quadratic programming problems in Mathematica, also manually.

4.1 Recommendation

Based on the findings, it is recommended that:

i. Researchers and practitioners in optimization, operations research, engineering, and finance consider integrating ICQPP-RMA into their computational toolkits for solving inequality-constrained quadratic programming problems.
ii. The method’s minimal computational cost, combined with its accuracy and simplicity, makes it especially suitable for real-time applications, embedded systems, or cases involving repeated optimization cycles.
iii. Dantzig-Wolfe may remain a preferred method when problem structure aligns with block decomposition, though its implementation is more involved.

4.2 Suggestions for Further Studies

i. Future studies should explore the scalability and robustness of ICQPP-RMA when applied to higher-dimensional quadratic programming problems, including those with hundreds or thousands of variables and constraints.
ii. Comparative performance should also be tested across different computing environments and programming languages (e.g., RStudio, Python, MATLAB, Julia) to assess portability and generalization.
iii. Research can extend the ICQPP-RMA framework to handle non-convex quadratic programming, quadratic integer programming, and multi-objective quadratic optimization.
iv. Further studies should integrate this method into commercial optimization libraries or solvers and assessing its performance on real-world datasets would further validate its practical relevance and impact.

References

[1] C. Ogboi, O. Ogunwale, J. O. Ogunwole, and N. B. Emordi, "Application of linear programming model in investment portfolio and loan portfolio optimization", *International Journal of Economics, Business and Management Research*, vol. 9, no. 5, pp. 447–463, 2025, doi: 10.51505/IJEBMR.2025.9529.
[2] H. Yue and P. Shen, "Bi-affine scaling iterative method for convex quadratic programming with bound constraints", *Mathematics and Computers in Simulation*, vol. 226, pp. 373–382, 2024, https://doi.org/10.1016/j.matcom.2024.01.018.
[3] M. K. Sah, N. Varma, and S. Kumar, "Quadratic programming problems and its solution techniques with neural network modeling", *Optimal Control Applications and Methods*, vol. 46, no. 2, pp. 766–774, 2024, https://doi.org/10.1002/oca.3042.
[4] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs", *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020, https://doi.org/10.1007/s12532-020-00179-2.
[5] M. Kılınç and F. Silva, "A review of quadratic programming applications in engineering and economics", *International Journal of Applied Mathematics and Computer Science*, vol. 30, no. 4, pp. 581–597, 2020, https://doi.org/10.34768/amcs-2020-0042.
[6] K. Vogklis and I. E. Lagaris, "An active-set algorithm for convex quadratic programming subject to box constraints with applications in non-linear optimization and machine learning", *Mathematics*, vol. 13, no. 9, p. 1467, 2025, https://doi.org/10.3390/math13091467.
[7] G. Grisetti, T. Guadagnino, I. Aloise, M. Colosi, B. Della Corte, and D. Schlegel, "Least Squares Optimization: From Theory to Practice", *Robotics*, vol. 9, no. 3, p. 51, 2020, https://doi.org/10.3390/robotics9030051.
[8] W. L. Winston, *Operations Research: Applications and Algorithms*, 5th ed., Cengage Learning, Boston, MA, 2023, ISBN 978-0-357-70042-9.
[9] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 11th ed., McGraw-Hill Education, New York, NY, 2021, ISBN 978-1-260-57910-5.

[10] J. Opara, C. J. Ogbonna, S. O. Ihekuna, and C. C. Ogbonna, "Proposed piecewise linear approximation for solving nonlinear transportation problems", in *International Journal of Mathematics Trends and Technology (IJMTT)*, vol. 65, no. 6, pp. 135–141, 2019, DOI n/a, URL available: ijmttjournal.org.
[11] S. C. Inyama, *Operation Research: Introduction*, Supreme Publisher, Owerri, 2007, ISBN 978-37734-5-7.
[12] P. K. Gupta and D. S. Hira, *Operations Research*, S. Chand & Company Ltd., New Delhi, 2011, DOI n/a, URL available: schandgroup.com.