MIE 335 Lab Project

Isis Gutierrez	Tugce Karatas	Doruk Unal
999079107	1001078433	998920397

April 4th 2014

Contents

1 Introduction

1.1 Problem

Our team has been tasked with optimizing a quadratic programming (QP) problem of the following form:

$$\begin{aligned} \text{minimize } & \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{subject to } & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{QP}$$

where x is a vector of continuous decision variables and Hessian H is positive semi-definite.

The team has applied three algorithms to this quadratic programming problem in hopes of arriving at an approximately optimal solution with a reasonable computation time. In this report, the team will predict which algorithm will perform the best based on criteria discussed in Section 1.3 and determine which algorithm performed the best in application. The team selected to apply Hildreth's Algorithm, Conjugate Gradient Method, and the Simplex Method for Quadratic Programming.

1.2 Algorithms

Hildreth's Algorithm was chosen based it's application of the primal-dual concept. The ability to solve the dual, a computationally easier problem, was a key factor in the appeal of this algorithm. Hildreth's algorithm is an iterative process and looks at each matrix element individually. Therefore, there is no matrix inversion required for this problem, an operation that is computationally expensive. [3]

The Conjugate Gradient Method is a method for solving systems of linear equations using gradients. It is an iterative algorithm whose search is based on solving the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ with search directions chosen by the conjugation of the residuals. This method is chosen because one will arrive at approximately the optimal solution regardless of the starting point. The Conjugate Gradient Method has a simple formula for finding a new unique direction. [2]

The Simplex Method for Quadratic Programming is a method that solves quadratic programs by using simple operations in the Simplex Tableau. In this algorithm, the main matrix is formed by combining all other matrices, resulting in an organized problem. Using this tableau, all calculations and row operations can be performed. This method iterates by always improving the optimal objective function value. It is accurate in finding the optimal objective. This method was chosen based on its organizational implementation. [1]

1.3 Criteria

The algorithms are evaluated on multiple criteria that collectively attempt to determine which algorithm is the most suitable to solve this quadratic program. The criteria are evaluated using quantitative metrics and were chosen because together they give a comprehensive picture of the quality of the algorithm. The criteria used ranged from computational speed to number of constraints violated. The following lists the criteria used for evaluating the algorithms:

1. Time Comparison: The average computation time is recorded for all three algorithms using the same data set and a variety of matrix dimensions $(n \times n)$. The best algorithm for this criteria will have the fastest average computational time.

2. Optimal Objective Function Value: The objective function value for all three algorithms using the same data set is compared. Because this is a minimization problem, the best algorithm will have the lowest objective function value.

3. Number of Non-Negativity Constraints Violated: The $\mathbf{x}$ values determined at the final solution may violate one or more non-negativity constraints, making the solution infeasible. The best algorithm in this case would adhere to all the non-negativity constraints, making the problem feasible. If more than one algorithm adheres to all the constraints, they are deemed to be ranked equal.

4. Number of Linear Constraints Violated: Similar to the non-negativity constraints, a solution is deemed infeasible if one or more of the linear constraints are violated. Therefore, the algorithms are evaluated on the number of linear constraints they adhere to. If more than one algorithm adheres to all the constraints, they are deemed to be ranked equal.

5. Optimal Objective Function Value Deviation: For each data set, a solution was found using all three algorithms as well as MATLAB's built in function quadprog. The % difference between the implemented algorithms and quadprog are measured. The algorithm with the smallest deviation from quadprog is deemed to be the best algorithm.

As one can see, there are many criteria that can be used to evaluate the algorithms. These criteria can be loosely grouped into time related criteria and accuracy related criteria. The best algorithm would ideally excel in both categories. However, in real world applications, the speed-accuracy trade-off effects algorithms. Determining a balance between having a computationally efficient algorithm with enough accuracy is the goal. Our final evaluation of the implemented algorithms will take this trade-off into effect. Our team has deemed feasibility to have a larger weight then computation time because it takes into account the integrity of the solutions. In this application, we have deemed a small level of infeasbility to be negliable. For non-negativity constraints, the accepted level of infeasibility is $\mathbf{x} \geq -1$.

2 Hildreth's Quadratic Programming Algorithm

2.1 Overview

Hildreth's Algorithm is built by applying an iteratively generated Lagrange multiplier λ when solving a system of equations. Lagrange multipliers is a strategy used for finding minima and maxima of a function subject to equality constraints. In this case, we have inequality constraints and therefore a variation to the method of Lagrange multipliers is used. Due to the fact that there are inequality constraints, we take into account that some constraints may be active while others will remain inactive at the optimal solution. An inequality constraint is active if the constraint is binding ($\mathbf{Ax} = \mathbf{b}$) and inactive if there is slack in the constraint. Kuhn-Tucker conditions are introduced into the algorithm to account for the variation in the method of Lagrange multipliers. These conditions say that $\lambda_i \geq 0$ if i is an active constraint and $\lambda_i = 0$ if i is inactive. The Kuhn-Tucker equations are:

$$\begin{aligned} Hx + c + \sum_{iS_{act}} A_i^T \lambda_i = 0 \qquad & \text{(KT)} \\ A_i x - b_i = 0 \quad i \in S_{act} \\ A_i x - b_i < 0 \quad i \notin S_{act} \end{aligned}$$

where S_{act} denotes the active set of constraints.

Hildreth's algorithm uses the concept of primal and dual problems. The active constraints belong to the primal problem. The Lagrange multiplier are the dual variables and can be used to identify the inactive constraints in the model. The primal problem is defined as:

$$max_{\lambda \geq 0} min_x [\frac{1}{2} x^T H x + x^T c + \lambda^T (Ax - b)]. \tag{1}$$

The dual problem is then defined using the matrices D and K calculated as

$$D = AH^{-1}A^T \tag{2}$$

$$K = b + AH^{-1}c. \tag{3}$$

Finally, the dual problem is denoted by

$$min_{\lambda \geq 0} (\frac{1}{2} \lambda^T D \lambda + \lambda^T K + \frac{1}{2} b^T H^{-1} b). \tag{4}$$

Once the optimal set of Lagrange multipliers λ_{act} are calculated, the primal variables can be calculated from the following equation

$$x = -H^{-1}c - H^{-1}A_{act}^T \lambda_{act} \tag{5}$$

where the active constraints A_{act} are the constraints that correspond to the active lambda values.

Hildreth used these concepts to develop an algorithm for solving quadratic programs, specifically the dual problem (equation 4). The algorithm starts with the Lagrange multipliers all equal to 0. The λ_i values are calculated one at a time to minimize the objective function (equation 4). If the calculated $\lambda_i < 0$, λ_i is set to 0, which decreases the objective function value. This is an iterative process that is terminated by $\lambda^m = \lambda^{m+1}$, where m is the iteration number. The process for calculating the lambdas is as follows:

$$w_i^{m+1} = \frac{-1}{d_{ii}}[k_i + \sum_{j=1}^{i-1} d_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} d_{ij}\lambda_j^m] \tag{6}$$

$$\lambda_i^{m+1} = max(0, w_i^{m+1}) \tag{7}$$

where d_{ij} is the $iijth$ element of the matrix D and k_i is the ith element of the vector K.

2.2 Pseudocode

The pseudocode for Hildreth's Quadratic Programming can be found in Algorithm 1. The pseudocode was developed by working through the problem 2.10 of Model Predictive Control System Design and Implementation Using MATLAB, a basic quadratic program solved by hand using Hildreth's Algorithm. [3]

Algorithm 1 Hildreth's Algorithm

$N =$ size of $\mathbf{b}$
$\mathbf{x}_0 = -\mathbf{H}^{-1}\mathbf{c}$
if $\mathbf{Ax} \leq \mathbf{b}$ **then**
    return $\mathbf{x}_0$
$\mathbf{d} = \mathbf{AH}^{-1}\mathbf{A}^T$
$\mathbf{k} = \mathbf{b} + \mathbf{AH}^{-1}\mathbf{c}$
$\lambda^0 = \mathbf{0}$
while $\lambda^{m+1} \neq \lambda^m$ **do**
    for $i = 1...N$ **do**
        $w_i^{m+1} = \frac{-1}{d_{ii}}[k_i + \sum_{j=1}^{i-1} d_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} d_{ij}\lambda_j^m]$
        $\lambda_i^{m+1} = max(0, w_i^{m+1})$
$\mathbf{x} = \mathbf{x}_0 - H^{-1}A^T\lambda^{final}$
$z = \frac{1}{2}\mathbf{x}^T\mathbf{Hx} + \mathbf{c}^T\mathbf{x}$
return $\mathbf{x}, z$

2.3 Complexity Analysis

The time complexity analysis for Hildreth's Quadratic Programming Algorithm is seen in Table 1. The complexity of the steps outside the loop were not considered in this calculation due to the fact that the number of steps would be

dominated by the loop statements. Two version of this algorithm were considered when developing the MATLAB code. The first version of the algorithm contained a maximum number of iterations that could be performed before returning the solution vector x. The second version contained no maximum iteration stopping condition and the algorithm completed when $\lambda_{old} = \lambda_{new}$. Ultimately, the second version of algorithm was used because when using a maximum number of iteration the feasiblity was often violated. Therefore, in the worse case scenario, the while loop will run approximately an infinite number of times, which will be denoted by the constant K. The complexity of this algorithm is as follows

$$O(n) \approx K[3 \times n \times n] \approx Kn^2 \tag{8}$$

where n is equal to the number of constraints.

The space complexity analysis of Hildreth's Quadratic Programming Method is seen in Table 2. By summing up the number of bits required to perform the algorithm, we see that the complexity is as follows

$$O_n \approx 64mn + 64m + 64n^2 + 64n + 64n + 8n + 64m + 64n + 64n \tag{9}$$

$$\approx 264n + 64mn + 64n^2 + 64m \tag{10}$$

$$O(n) \approx n^2 + mn \tag{11}$$

where m is equal to the number of constraints and n is equal to the number of decision variables. With this data generator, $m \geq n$.

Functions	Numbers of Calls	Complexity
$w_i^{m+1} = \frac{-1}{d_{ii}}[k_i + \sum_{j=1}^{i-1} d_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} d_{ij}\lambda_j^m]$	n	$2n + (n-1) + 4 \approx 3n$
$\lambda_i^{m+1} = max(0, w_i^{m+1})$	n	$1 \times n$

Table 1: Time complexity of Hildreth's Quadratic Programming Algorithm

Variable	Type	Dimensions	Bits Required
A	double	$m \times n$	$64mn$
b	double	$1 \times m$	$64m$
H	double	$n \times n$	$64n^2$
x_0	double	$1 \times n$	$64n$
C	logical	$1 \times m$	$8n$
d	double	$m \times m$	$64m^2$
k	double	$1 \times m$	$64m$
λ_{old}	double	$1 \times n$	$64n$
λ_{new}	double	$1 \times n$	$64n$

Table 2: Space complexity of Hildreth's Quadratic Programming Algorithm

2.4 Performance Predictions

There are factors that benefit and that hinder the performance of Hildreth's algorithm. Hildreth's algorithm is an iterative process and looks at each matrix element individually. Therefore, there is no matrix inversion required for this problem, an operation that is computationally expensive. The constrasting point is that with large matrices, looking at each element individually will be a time consuming task. [3]

There are two possible stopping conditions to Hildreth's algorithm. The first that the Lagrange multipler will converge. This will only happen if the active constraints are linearly indepenident and the number of active constraint is less than or equal to the number of decision variables (size of vector x). Therefore a tolerance is added of $+/-1e3$. Otherwise, the other option was having the algorithm terminate when a predefined number of iterations have been completed. This option was deemed to be better in terms of time complexity, however through testing it was determine that having a maximum number of iteration impacted the feasibility of the problem. Therefore, in the application of the algorithm the first stopping condition was used. It is predicted that this will severely impact the computational time performance of the algorithm and that this algorithm will be the slowest of the three. However, it is also predicted that this algorithm will return a feasible solution within 5% of optimality.

3 Conjugate Gradient Method

3.1 Overview

Conjugate Gradient Method implements the method of steepest descent in solving systems of linear equations. It solves $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a symmetric positive definite matrix, in n iterations. The algorithms iteratively performs a line search in unique search directions to arrive at the optimal solution.

Conjugate Gradient Method is based on the set of A-orthogonal search directions. In each iteration, the direction is chosen by the conjugation of the residuals. Residuals are orthogonal to previous search directions and therefore until the residuals are equal to zero, linearly independent directions are produced. When Conjugate Gradient Methods for quadratic functions such as $z(x) = \frac{1}{2}\mathbf{x}^T\mathbf{Hx} + \mathbf{c}^T\mathbf{x}$, solving this function gives exactly the same solution to the equation Ax+b=0 because this equation is the gradient of the objective function.

There are several assumption that were made for Conjugate Gradient Method in this application [2]:

1. The initial point was chosen to be $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, which is a point that satisfies the linear constraints. This method deals with equality constraints and therefore using a starting point enables the algorithm to reach the optimal solution in generally less than n iterations.

2. The Conjugate Gradient Method uses symmetric positive definite (SPD) matrices. However, the $\mathbf{A}$ matrix used in this problem is a semi-positive definite matrix. In order use the $\mathbf{A}$ matrix, it will be transformed $\mathbf{A} = \mathbf{A}^T\mathbf{A}$. This will provide a SPD matrix with dimensions consistent with the $\mathbf{H}$ matrix.

3. Normally, the Conjugate Gradient Method will iterate until the gradient of the objective function is zero. The unconstrained solution to the minimization of the objective function $z(x) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x}$ is equal to $\mathbf{x} = H^{-1}c$. Because this is a constrained problem, it is assumed that $\mathbf{A}\mathbf{x} = \mathbf{b}$ needs to be solved. The following calculation were performed to determine the residuals

$$\mathbf{H}\mathbf{x} + \mathbf{c} = \mathbf{0} \tag{12}$$

$$\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T b = \mathbf{0} \tag{13}$$

$$(\mathbf{H} + \mathbf{A}^T\mathbf{A})\mathbf{x} + \mathbf{c} - \mathbf{A}^T\mathbf{b} = \mathbf{0} \tag{14}$$

$$\mathbf{A}^T\mathbf{b} - \mathbf{c} - (\mathbf{H} + \mathbf{A}^T\mathbf{A})\mathbf{x} = r. \tag{15}$$

4. In order to eliminate floating rounding errors, this method uses a different residual formula for iterations number i where the $i \bmod 50 = 0$.

5. In order to eliminate the possible increase in objective function value, $z_{new} - z_{old}$ is calculated and used as a stopping criteria for the loop.

6. The β value used in the algorithm should be positive. This results in the algorithm continuing while stepping outside the feasible space (because this is a constrained version of the algorithm). However, in this application a negative β value is used in order to remain within the constrained space. The downside is that this β value allows the objective function value to increase with steps taken and was mediated by the previous assumption.

3.2 Pseudocode

The pseudocode for Conjugent Gradient Method can be found in Algorithm 2. The pseudocode was developed by analysing and applying the pseudocode found in the paper "An introduction to the conjugate gradient method without the agonizing pain" and making the necessary alterations for constrained problems. [2]

3.3 Complexity Analysis

The time complexity analysis for Conjugate Gradient Method is seen in Table 3. The complexity of the algorithm relies on the number of variables, the number of constraints, and the number of iterations needed to arrive at the optimal

Algorithm 2 Conjugate Gradient Method

$x_0 \leftarrow A^{-1}b$
$res \leftarrow A^Tb - c - (H + A^TA)x$
$res_{new} \leftarrow res$
$q_{new} \leftarrow res^T res$
$q_0 = q_{new}$
$z_{new} \leftarrow z(x)$
$z_{old} \leftarrow 2z_{new}$
while $q_{new} > tolerance^2 q_0$ & $z_{new} < z_{old} + tolerance$ & $n < nmax$ **do**
    $value \leftarrow (H + A^TA + x^Tx)res_{new}$
    $\alpha \leftarrow \frac{q_{new}}{resnew^T value}$
    $x \leftarrow x + \alpha res_{new}$
    $x_0 \leftarrow x - \alpha res_{new}$
    $q_{old} = res^T res$
    if $n \bmod 50 == 0$ **then**
        $res \leftarrow res + A^Tb - c - (H + A^TA)x$
    else
        $res \leftarrow res - \alpha value$
        $q_{new} = res^T res$
        $\beta \leftarrow \frac{-q_{new}}{q_{old}}$
        $res_{new} \leftarrow res + \beta res_{new}$
        $n = n + 1$
        $z_{old} \leftarrow z_{new}$
        $z_{new} \leftarrow \frac{1}{2}x^THx + c^Tx$
$x \leftarrow \frac{(x+x_0)}{2}$
return x

solution. In the worst case scenario, the number of iterations $numiter = n$. The complexity of this algorithm is as follows

$$O_n \approx 4mn^2 + n^3 + 4mn + 6n^2 + 5n - 2 + n(2n^3 + 5n^2 + 16n - 3) \quad (16)$$

$$O(n) \approx mn^2 + n^4 \quad (17)$$

where m is equal to the number of constraints and n is equal to the number of decision variables.

Functions	Numbers of Calls	Complexity
$x = A\backslash$ b	1	$(n^2+n)(2m-1)+n^3$
$r =$A$^\top$ b-Hx-A$^\top Ax$	1	$2mn + 2mn^2$
$q_{new} =$r$^\top$ r	1	$2n-1$
$z_{new} = 0.5$X$^\top$Hx+c$^\top x$	1	$(2n-1)(n+1)$
$z_{old} = 2z_{new}$	1	1
$value = (H+$A$^\top$ A+x$^\top x)resnew$	numiter	$(n^2+n+1)(2n-1)+2n^2$
$\alpha = q_{new}/($r$^\top_{new} value)$	numiter	$2n$
$x = x + \alpha r_{new}$	numiter	$2n$
$x_0 = x - \alpha r_{new}$	numiter	$2n$
$q_{old} =$r$^\top$ r	numiter	$2n-1$
$r = r - \alpha value$	numiter	$2n$
$q_{new} =$r$^\top$ r	numiter	$2n-1$
$\beta = -q_{new}/q_{old}$	numiter	1
$r_{new} = r + \beta r_{new}$	numiter	$2n$
$z_{new} = 0.5$X$^\top$Hx+c$^\top x$	numiter	$(2n-1)(n+1)$
$x = (x + x_0)/2$	1	$2n$
$z = 0.5$X$^\top$Hx+c$^\top x$	1	$(2n-1)(n+1)$

Table 3: Time complexity of Conjugate Gradient method

The space complexity analysis of Conjugate Gradient Method is seen in Table 4. By summing up the number of bits required to perform the algorithm, we see that the complexity is as follows:

$$O_n \approx 64mn + 64n^2 + 64m + 384n + 640 \quad (18)$$

$$O(n) \approx mn \quad (19)$$

where m is equal to the number of constraints and n is equal to the number of decision variables.

Variable	Type	Dimensions	Bits Required
A	double	$m \times n$	$64mn$
b	double	$1 \times m$	$64m$
H	double	$n \times n$	$64n^2$
x	double	$1 \times n$	$64n$
x_0	double	$1 \times n$	$64n$
c	double	$1 \times n$	$64n$
n_{max}	single	1×1	32
ϵ	double	1×1	64
r	double	$1 \times n$	$64n$
r_{new}	double	$1 \times n$	$64n$
q_{new}	double	1×1	64
q_0	double	1×1	64
q_{old}	double	1×1	64
$value$	double	$1 \times n$	$64n$
$numiter$	single	1×1	32
z_{new}	double	1×1	64
z_{old}	double	1×1	64
z	double	1×1	64
α	double	1×1	64
β	double	1×1	64

Table 4: Space complexity of Conjugate Gradient

3.4 Performance Predictions

This method was chosen because the formula to calculate the new direction is simple. However, there are some pitfalls to this algorithm that will effect the performance. Although generally it takes n step to reach an optimum solution, because of the round-off error it can require more than n steps. The opposite side of this argument is that if the initial point is chosen as an estimated point, generally it takes less than n steps to reach to the optimum solution. [2]

Furthermore, this algorithm works is computationally attractive for unconstrained large scale systems, but when constraints are included the implementation become difficult. After implementing the numerous assumptions to the unconstrained algorithm in order to satisfy all the constraints, the computational attractiveness is reduced.

It is predicted that this method will be relatively faster than Hildreth's method because there will always be a maximum number of iterations. Because we are starting with a feasible solution, it is also predicted that the algorithm will not need to complete the maximum number of iterations. The algorithm should reach within 5% optimality and remain within the feasible region.

4 Simplex Method for Quadratic Programming

4.1 Overview

The Simplex Method for Quadratic Programming is based on finding the local minimum when the objective function is convex for all feasible solutions. It can be applied once the Karush-Kuhn-Tucker (KKT) conditions are constructed. The Lagrangian function is used to to find the necessary KKT conditions:

$$L(x, \mu) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x} + \mu(\mathbf{A}\mathbf{x} - \mathbf{b}) \tag{L}$$

where the vector μ defines the optimal dual variables.

After introducing non-negative surplus variables and non-negative slack variables, the final form of the KKT conditions are written as:

$$\mathbf{H}\mathbf{x} + \mathbf{A}^T\mu^T - y = -\mathbf{c}^T \tag{20}$$

$$\mathbf{A}\mathbf{x} + v = \mathbf{b} \tag{21}$$

$$\mathbf{x} \geq 0, \mu \geq 0, y \geq 0, v \geq 0 \tag{22}$$

$$y^T\mathbf{x} = 0, \mu v = 0. \tag{23}$$

The optimal solution can be found by introducing artificial variables to each equation and minimizing the sum of these artificial variables. The following equations can now be put into simplex form to be solved

$$\mathbf{H}\mathbf{x} + \mathbf{A}^T\mu^T - y + a_1 = -\mathbf{c}^T \tag{24}$$

$$\mathbf{A}\mathbf{x} + v + a_2 = \mathbf{b} \tag{25}$$

$$\mathbf{x} \geq 0, \mu \geq 0, y \geq 0, v \geq 0 \tag{26}$$

$$y^T\mathbf{x} = 0, \mu v = 0. \tag{27}$$

In each iteration of the simplex method, the entering variable is denoted as the variable not in the current basis that has the least of the reduced costs, provided that its complement is not in the basis or would take the place of its complement. The departing variable is denoted by the variable whose right hand side of the equation in divided by the coefficient of the entering variable is the smallest. After the entering and departing variable are found they switch places and iteration continues by calculating the new reduced costs of the variables that are not in the basis.

4.2 Pseudocode

The pseudocode for The Simplex Method for Quadratic Programming can be found in Algorithm 3. The pseudocode was developed by working through the example in Operations Research Models and Methods, Nonlinear Complex Methods.S2, a basic quadratic program solved by hand using The Simplex Method. [1]

Algorithm 3 The Simplex Method for Quadratic Programming

1: Construct the Simplex Tableau:

$$ST = \begin{bmatrix} \mathbf{H} & \mathbf{A}^T & -I & 0 \\ \\ \mathbf{A} & 0 & 0 & I \end{bmatrix}$$

2: For any $RHS_i < 0$, do: $row_i = -row_i$

3: Add artificial variables, update ST:

$$ST = \begin{bmatrix} \mathbf{H} & \mathbf{A}^T & -I & 0 & I & 0 \\ \mathbf{A} & 0 & 0 & I & 0 & I \end{bmatrix}$$

4: Calculate reduced costs $r = (-1) \sum_{i=1}^{size(H)+size(A)} row_i$

5: **while** $\min\{r\} < 0$ **do**

6: find entering variable $e = \arg min_{j \in ST}\{r\}$

7: find departing variable $d = \arg min_{j \in ST}\{RHS./ST(index\ of\ e)\}$ where $ST(index\ of\ e) > 0$

8: Check if constraints apply: $\mu v = 0, \mathbf{x}y = 0$

9: Modify ST by making $ST(index\ of\ min\_ratio,\ index\ of\ e) = 1$ and the rest of the column $= 0$ using row operations

10: $z = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{c}^T\mathbf{x}$

11: return $\mathbf{x}, z$

4.3 Complexity Analysis

The time complexity of the simplex method can be seen in Table 5. The complexity of this algorithm was calculated by determining the number of multiplications and additions that happen in the Simplex Tableau. It is coded as if the quadratic programming problem is solved by hand. Therefore, the algorithm is executed by only modifying the Simplex Tableau. The time complexity of the Simplex method coded in MATLAB is as follows:

$$O_n \approx 4m^2 + 6n^2 + 8mn + 2m + 3n - 1 + (n+m)^{n+m}(3m^2 + 3n^2 + 6mn + 2m + n + 2) \tag{28}$$

$$O(n) \approx (n+m)^{(n+m+2)} \tag{29}$$

where m is equal to the number of constraints and n is equal to the number of decision variables.

Functions	Numbers of Calls	Complexity
Find reduced costs	1	$(2m+2n)(n+m)+(2m+2n)$
Make RHS positive	1	$(n+m)(2n+2m)$
Find Minimum ratio	numiter	$n+m$
Constraint check ($\mu v = 0, \mathbf{x}y = 0$)	numiter	m
Set coeff of departing variable = 1	numiter	$(3m+3n+1)$
Row operations	numiter	$(3m+3n+1)(n+m-1)$
Switch entering, departing variable	numiter	2
$z = 0.5\mathrm{X}^\top \mathrm{Hx} + \mathrm{c}^\top x$	1	$(2n-1)(n+1)$

Table 5: Time complexity of Simplex Method

The Space complexity of the Simplex Method is calculated by determining the total number of bits in the decision variables and the entirety of the Simplex Tableau. These numbers can be found in table 6. The space complexity of the Simplex method is as follows:

$$O_n \approx 64m^2n + 192n^2 + 128m^2 + 192mn + 576n + 576m + 64 \tag{30}$$

$$O(n) \approx m^2n + n^2 \tag{31}$$

where m is equal to the number of constraints and n is equal to the number of decision variables.

Variable	Type	Dimensions	Bits Required
A	double	$m \times n$	$64mn$
b	double	$1 \times m$	$64m$
H	double	$n \times n$	$64n^2$
x	double	$1 \times n$	$64n$
α	double	$(n+m) \times (n+m)$	$64(n+m)(n+m)$
μ	double	$(m+n) \times m$	$64m^2n$
y	double	$(m+n) \times n$	$64n(n+m)$
v	double	$(m+n) \times m$	$64m(m+n)$
c	double	$1 \times n$	$64n$
r	double	$1 \times (3n+3m)$	$64(3n+3m)$
B	double	$1 \times (n+m)$	$64(n+m)$
N	double	$1 \times (3m+3n)$	$64(3m+3n)$
z	double	1×1	64

Table 6: Space complexity of Simplex Method

4.4 Performance Predictions

The Simplex method was chosen because with each iteration it selects an entering variable that has the possibility of having the most impact on the current optimal objective function value. However, since the Simplex Method does not stop looking for the optimal solution until all of the reduced costs are zero, the number of iterations it goes through may increase rapidly as the dimension of the Hessian matrix and the number of decision variables increase.

The complexity analysis performed on this method shows that the time it takes to solve a quadratic programming problem increases exponentially. Thus, the Simplex Method will not be efficient solve problems with extremely large values of n. When comparing complexities, it was expected that the Simplex method would take less time than Hildreth's algorithm, but more time than the Conjugate Gradient method.

It was also predicted that most of the time the optimal objective function value found using the Simplex Method would be within 5% range of the actual optimal objective found by MATLAB's Quadprog function. However, the occurrence of outliers was also expected as the Simplex method updates its optimal solution basis according to the existing non-basic variables specific to each iteration. Therefore, it is probable that the Simplex method may find an optimal objective deeply out of range.

5 Results

Once the algorithms were created using MATLAB, performance measurements were taken. For all the measurements in this section, for each trial, the same data set was used to calculate the values. This allows for the direct comparison

of the metrics measured. The data sets were generated using the provided dataGenerator function. For each trial, a new data set was generated.

Ideally, these measurement would be taken for $n \times n$ matrices (the dimension of $\mathbf{H}$ and the number of variables in the original problem) from $n = 1..X$, where X is a large number such as 1000. However, for the sake of time and due to computational power constraints, the evaluation metrics were only taken from $n = 1..40$. To determine the overall measurement, the average of 3 trials is taken. Hildreth's Method was the constraining algorithm, as the maximum number of iterations was not fixed and therefore the time to compute values was significant.

The first criteria analysed is average computational time and is plotted on Figure 1. The graph displays the dimension n versus the average computational time in seconds. The graph clearly displays that Hildreth's is the worse in terms of computational time, which follows the predictions. The average computational time for Hildreth's seems to follow no clear pattern. This may be because the data is randomly generated. Depending on the data set, the number of iterations required to reach the optimal solution from the starting point will vary widely from 1 to inifinity and therefore the trend is not clear. However, this graph displays a very small portion of the number of trials that could have been conducted. In order to make a more appropriate estimation of the computational time, more trials should be conducted for each n and the number of n values plotted should increase to give a larger scale picture of metric.

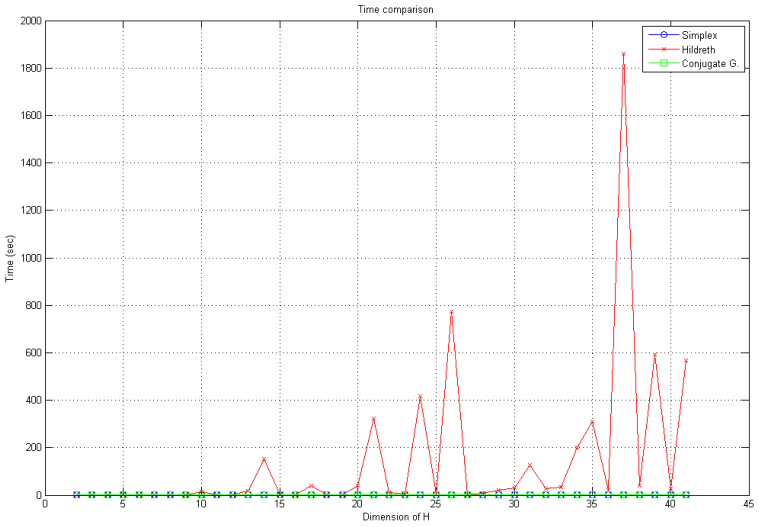

Figure 1: Average Computational Time

The trend for Simplex and Conjugate Gradient computation times is not visible in Figure 1 due to the fact they are dominated by the time of Hildreth's. Therefore, a second comparison is created to compare the average computational times of Simplex and Conjugate Gradient and is displayed as Figure 2. This graph shows a clear difference between the computational time of Simplex and Conjugate Gradient. The Simplex Method appears to be approximately increasing exponentially with the increasing matrix size n. However, to make a more accurate estimation, the number of n values plotted should increase to give a larger scale picture of the computational time. The Conjugate Gradient Method appears to be following a constant time. The constant is small and makes the algorithm very attractive. Overall, we have determined that Hildreth's algorithm is clearly the worst option in terms of computational time. From the number of values tested, we have determined that Conjugate Gradient performs the best in terms of computational time, however more trials need to be performed to reach a more accurate conclusion.

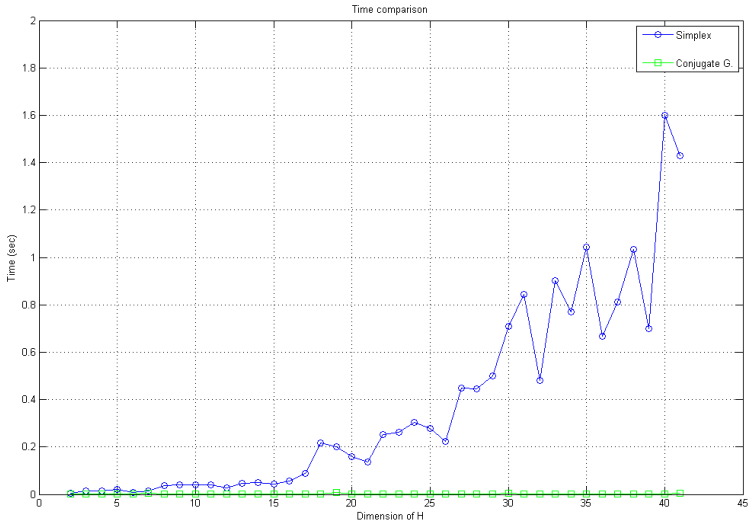

Figure 2: Average Computational Time of The Simplex Method and Conjugate Gradient Method

Optimal objective function calculated by each algorithm is compared on Figure 3. This graph shows that all the algorithms reach approximately the same optimal solution. As the size of n increases, we see a deviation between the optimal objective value computed. This is a minimization problem and therefore the best algorithm for this criteria will return the smallest objetive function value.

In this case, we see that Simplex Method and Hildreth's method are approximately the same and therefore are deemed as better than Conjugate Gradient Method. Because the values are very close, in order to make a more appropriate estimation on the optimal objective function value, more trials should be conducted for each n and the number of n values plotted should increase to give a larger scale picture.

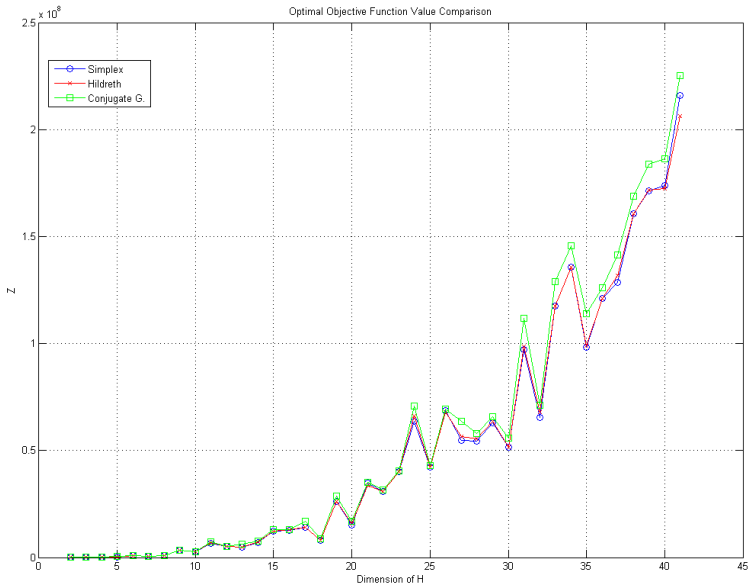

Figure 3: Comparison of Optimal Objective Function Value

The final criteria used for evaluating the objective function is the % deviation from the quadprog solution, which can be seen on Figure 4. As seen on the graph, Hildreth's algorithm has the smallest percent deviation from quadprog and therefore is the best algorithm based on this criteria. The next best algorithm is the Simplex Method which for the majority of the time is within 5% of the quadprog solution. The worst offender in the deviation category is Conjugate Gradient Method that violated the 5% tolerance limit by varying amounts. Therefore, the team looked the Conjugate Gradient Method MATLAB code to look for ways to improve the optimal objective function value. The code was edited to achieve an optimal objective function within 5%. A graph was generate for comparing quadprog and the updated Conjugate Gradient Method and can seen in Figure 5. This graph shows a drastic improvement in the objective function value. Figure 6 displays the updated Conjugate Gradient Method objective value in comparison with the objective value obtained using quadprog.

The number of linear constraints violated is important in evaluating the

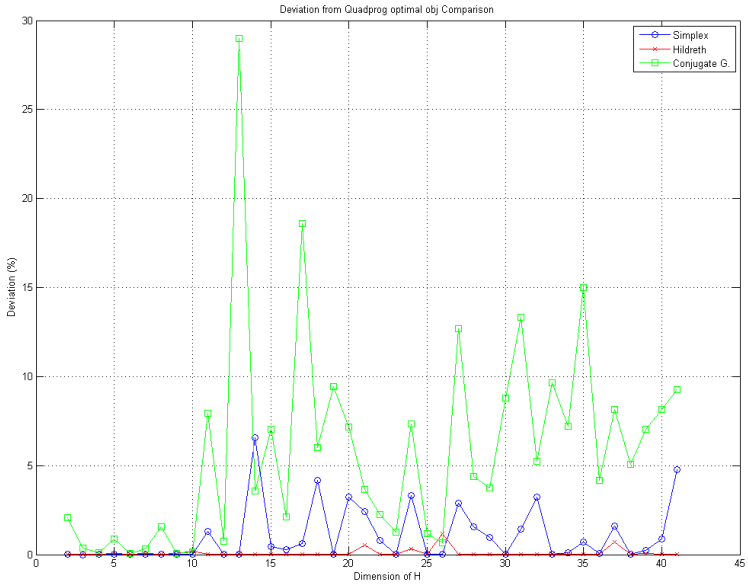

Figure 4: Deviation from Quadprog() solution

algorithms. If any linear constraints are violated, the solutions is deemed infeasible and the solution cannot be used. For the purpose of this project, small violations of the constraints are deemed acceptable and will not gravely impact the feasibility of the solution. The number of linear constraints violated for each algorithm can be seen in Figure 7. From this graph, we can see that Conjugate Gradient Method is the only algorithm that does not violate any of the linear constraints. Therefore, at first glance, this is the best algorithm for this criteria.

The second metric used to measure the linear constraints can be seen in Figure 8 which displays the maximum value when computing $\mathbf{Ax} = \mathbf{b}$ for all m constraints, where $\mathbf{x}$ is the optimal solution given by the algorithm. For a solution to be feasible, the value plotted must satisfy $\mathbf{Ax} \leq \mathbf{b}$ and therefore the value computed by solving the inequality must be ≤ 0 within some tolerance. The maximum violation was plotted for each trial to determine the severity of the violations and to see if the constraint violated fell within the tolerance, making it a feasible solution. From the graph, one can clearly see that the Simplex Method and Hildreth's Method violate constraints and therefore are not implementable solutions to this quadratic program. In terms of linear constraints, Conjugate Gradient Method is the best. There is no need to run more trials for this criteria as the trend is evident. In the future, the Simplex Algorithm and Hildreth's algorithms can be improved to violate less constraints. This will most likely increase the time component. To improve the number of violated

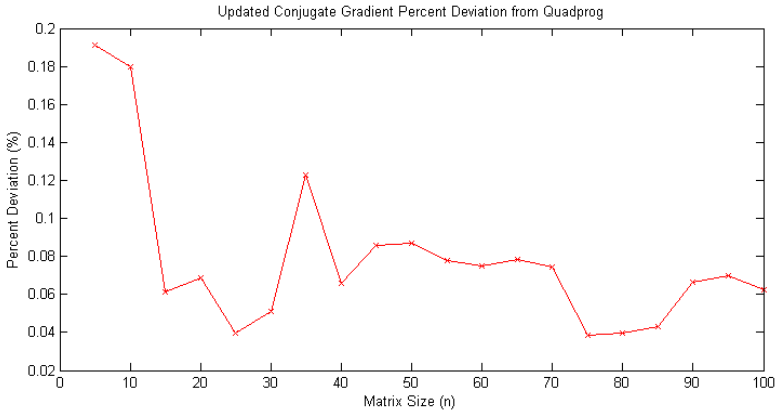

Figure 5: Updated Conjugate Gradient's Deviation from Quadprog() solution

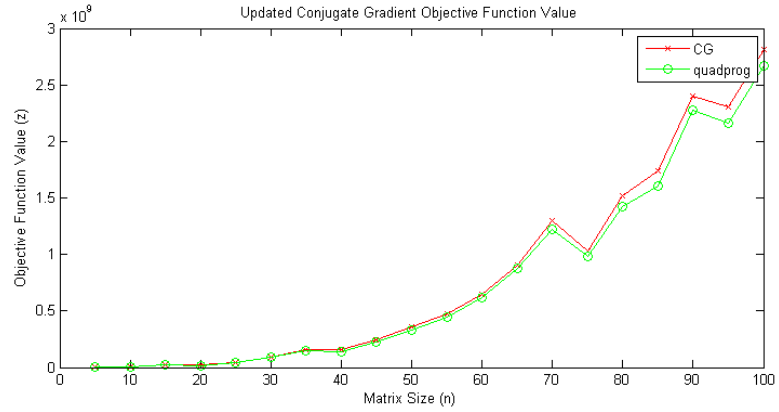

Figure 6: Deviation from Quadprog() solution

constraints for Hildreth's algorithm, the tolerance can be made smaller than $1e-3$. This will effect the computational time and make the algorithm less attractive. This is a key example of the speed accuracy trade off.

The number of non-negativity constraints violated is important in the evaluation the algorithms. If any non-negativity constraints are violated, the solutions is deemed infeasible and the solution cannot be used. For the purpose of this project, small violations of the constraints ($x \geq -1$) are deemed acceptable and will not gravely impact the feasibility of the solution. The number of non-negativity constraints violated for each algorithm can be seen in Figure 9. From this graph, we can see that Conjugate Gradient Method and the Simplex Method do not violate any of the non-negativity constraints. Therefore, in terms of this criteria, Conjugate Gradient Method and the Simplex Method are deemed equally the best. Hildreth's method can be improved to violate less

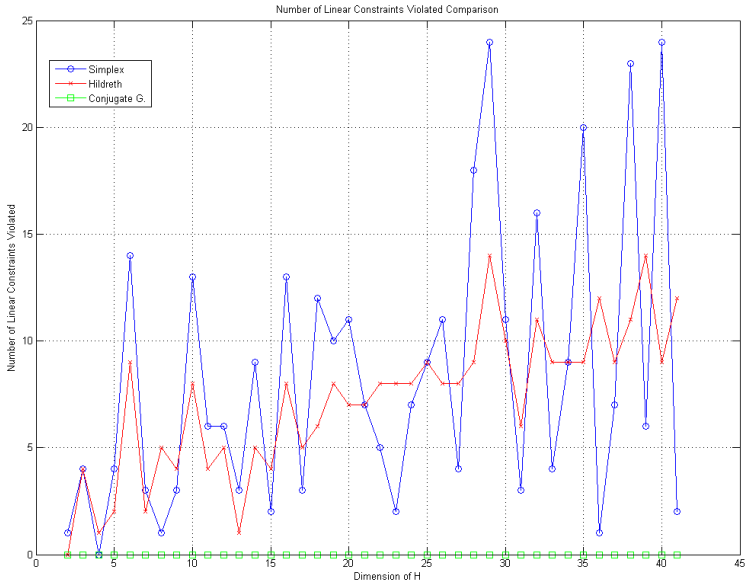

Figure 7: Number of Linear Constraints Violated

non-negativity constraints by making the tolerance can be made smaller than $1e-3$. This would impact the speed of the already slow algorithm.

The second metric used to measure the non-negativity constraints can be seen in Figure 10 which displays the minimum value $\mathbf{x}$ in the optimal solution given by the algorithm. The minimum violation was plotted for each trial to determine the severity of the violations and to see if the constraints violated fell within the tolerance, making it a feasible solution. From the graph, one can clearly see that the Hildreth's Method violates constraints and therefore is not an implementable solution to this quadratic program. In terms of non-negativity constraints, Conjugate Gradient Method and the Simplex Method are tied for the best. There is no need to run more trials for this criteria as the trend is evident.

6 Recommendation

Depending on the situation, the algorithm appropriate for usage will vary. In terms of a feasible solution and computational time, the algorithm that we have deemed to be the best is Conjugate Gradient Method. This is because this is the only algorithm that does not violate any constraints (non-negativity and linear) and has the shortest average computational time for all $n \times n$ sized matrices tested. The downside of using this algorithm is that had the largest percent

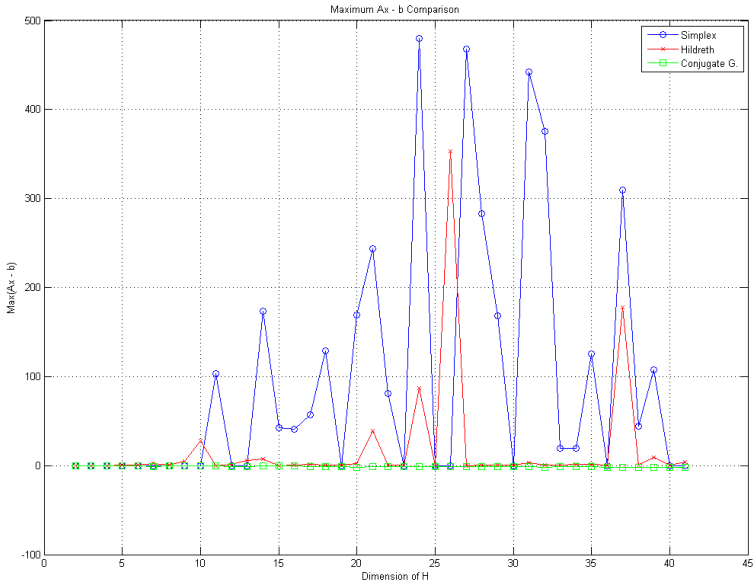

Figure 8: Maximum Ax-b Values

deviation from quadprog's optimal objective function value. We were then able to update the Conjugate Gradient Method MATLAB code in order to achieve the tolerable (5%) deviation from the optimal solution given by quadprog.

Each algorithm has positive and negatives. One needs to be wary of this when choosing this algorithm and must assure that they have enough computational power in order to solve this quadratic program in a reasonable amount of time. The analysis of Hildreth's algorithm, Conjugate Gradient Method, and the Simplex Method clearly display the speed accuracy trade off. When one is deciding on which algorithm to implement, they must be aware of this trade off and chose an algorithm based on their priorities.

References

[1] Jonathan F. Bard Paul A. Jensen. Nonlinear programming method.s2 quadratic programming. *Operations Research Models and Methods*, pages 1–5, n.d.

[2] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. *School of Computer Science Carnegie Mellon University*, pages 21,41,50, August 4, 1994.

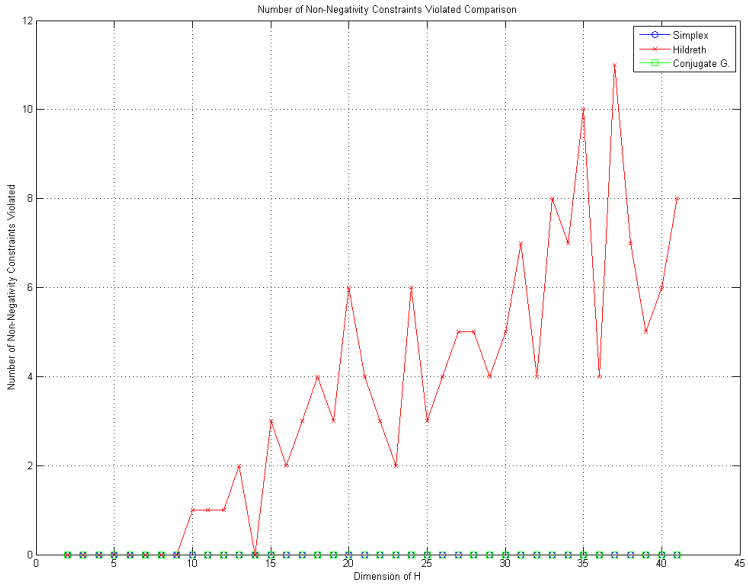

Figure 9: Number of Non-negativity constraints violated

[3] Liuping Wang. *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, Melbourne, Australia, 2009.

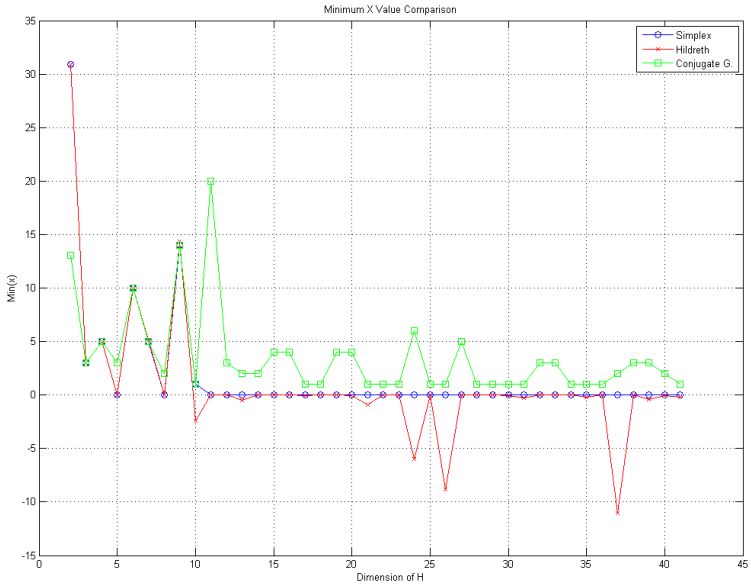

Figure 10: Minimum X Values