Sonny Chauhan
Final project

Topic 1: Support Vector Machines

I developed an ALM solver for the first problem. The subproblem that is solved within each outer iteration is

$$(1)$$

$$argmin_{w,b,t} L_\beta(w,b,t,u) \quad (2)$$

$$. \quad (3)$$

with the inequality constraint $t_i \geq 0$. When we minimize our augmented Lagrangian for a fixed u, that is solving the subproblem. The augmented lagrangian function also written as $L_\beta(w,b,t,u)$ like written above is $\sum_{i=1}^{n} t_i + \frac{\lambda}{2} * \|w\|^2 + \sum_{i=1}^{m} u_i * (1 - t_i - y_i * (\mathbf{w^\intercal x_i} + b)) + \frac{\beta}{2} * \max(0, 1 - t_i - y_i * (\mathbf{w^\intercal x_i} + b))^2$ where m is number of columns of X. I then vectorized the function $1 - t_i - y_i * (\mathbf{w^\intercal x_i} + b)$ for all $i = 1, ...m$ in my augmented Lagrangian function. I got a column vector where each row has components of this function such that the first row of this column vector has the entry $1 - t_1 - y_1 * (\mathbf{w^\intercal x_1} + b)$, the second row of this column vector has the entry $1 - t_2 - y_2 * (\mathbf{w^\intercal x_2} + b)$, ..., up until the mth row of the column vector has the entry $1 - t_m - y_m * (\mathbf{w^\intercal x_m} + b)$. In order to do this I first created a vector of ones that is legnth of t. Then I subtracted each component of the vector t to get the second component of the vector. The expression $w^\intercal \mathbf{X}$ gives a row vector in which the first column is $w^\intercal \mathbf{x_1}$ where x_1 is first column of X, the second column is $w^\intercal \mathbf{x_2}$ where x_2 is second column of X,...., up until the mth column which is $w^\intercal \mathbf{x_m}$ where x_m is the mth column of X. I took the transpose of $w^\intercal \mathbf{X}$ which is equivalent to $X^\intercal \mathbf{w}$ in order to get a column vector whose first row is $w^\intercal \mathbf{x_1}$, second row is $w^\intercal \mathbf{x_2}$, ..., mth row is $w^\intercal \mathbf{x_m}$. I then multiplied this vector componentwise by $-y$ to get third part of the expression for column vector. Lastly I multiplied -y and b to get the last part of expression for the column vector. The sum of all the components of t can be found by multiplying transpose of t by a vector of ones that is same legnth of t. The third part of the expression of the augmented lagrangian function $\sum_{i=1}^{m} u_i * (1 - t_i - y_i * (\mathbf{w^\intercal x_i} + b))$ can be rewritten as $u^\intercal (\mathbf{1 - t - y} .* (\mathbf{X^\intercal w}) - \mathbf{y} * \mathbf{b})$ where $.*$ represents componentwise multiplication. This is because I am multiplying each component of u with the vector and then adding them all together. The last part of the augmented lagrangian can be rewritten by $\frac{\beta}{2} * \|1 - t - y. * (X^\intercal \mathbf{w}) - y * b\|_+^2$, where we take the positive part of vector before taking the norm. The augmented lagrangian function was found by first taking the function f(w,b,t) to minimize, in this case $\sum_{i=1}^{n} t_i + \frac{\lambda}{2} * \|w\|^2$. Then the inequality constraint $y_i * (\mathbf{w^\intercal x_i} + b)) \geq 1 - t_i$ was rewritten to get it in the form where the inequality ≤ 0. Thus the new inequality constraint

became $1-t_i-y_i*(\mathbf{w}^\intercal\mathbf{x_i}+b))\leq 0$ for all $i=1,2.....,n$ where n is the number of columns of matrix X. The inequality constraint was then put to get it in the form of an augmented lagrangian function with quadratic penalty which corresponds to $L_\beta\left(w,b,t,u\right)$ written above. In order to solve the subproblem I used projected gradient method. The iterative update formula in the projected gradient method for w is found by taking w0(initial value of w) and then subtracting the stepsize alpha multiplied by the gradient of the augmented Lagrangian function with respect to w0 from w, w = w0 - (alpha * gradient of augmented Lagrangian with respect to w0). The iterative update formula in the projected gradient method for b is found by taking b0(initial value of b) and then subtracting the stepsize alpha multiplied by the gradient of the augmented Lagrangian function with respect to b0 from b, b = b0 - (alpha * gradient of augmented lagrangian with respect to b0). Since we still have the inequality constraint that $t_i\geq 0$, then the iterative update formula for t with respect to projected gradient method is found by taking t0 and then subtracting the stepsize alpha multiplied by the gradient of the augmented Lagrangian function with respect to t0 from t and then making sure that each component of the new vector t is greater than or equal to 0 t= max(0, t0 - (alpha * gradient of augmented lagrangian with respect to t0)) since all the indices of vector t have to be greater than or equal to 0 based on the inequality constraint $t_i\geq 0$ for all indices i in t. Also the gradient of our augmented lagrangian function with respect to w,b,and t is updated after every iteration in our ALM solver in addition to the gradient error . The gradient error is found by first taking the gradient of the augmented Lagrangian function with respect to w,b,and t. Then we add the norm of the gradient of w with the norm of the gradient of b with respect to the augmented lagrangian function. With respect to the gradient of t, first we find the indices of vector t that are greater than 0. Then we get a new vector $gradt(id1)$ that contains the vector gradient of t with respect to only those indices, and then I take the norm of this vector,$norm(gradt(id1))$. Afterwards we find the indices of vector t that are equal to 0, then we find a new vector $gradt(id2)$ that is the vector gradient of t with respect to only those indices mentioned above. Then I make sure that each component of gradt(id2) is less than or equal to 0 by doing $(min(0,gradt(id2)))$, afterwards I take the norm of this vector, $norm(min(0,(gradt(id2)))$. Then I add the values of $norm(gradt(id1))$ and $norm(min(0,(gradt(id2)))$ to the values of norm of gradient of w and the norm of the gradient of b with respect to the augmented lagrangian function to get the final value for gradient error. We also update the iteration number (subit) of the projected gradient method by one . In addition I also update the the iteration number(iter) of our ALM solver by one in each iteration of our ALM method. The value of our multiplier u is also updated through each iteration of the alm method. The value of u is first initialized to 0 in our alm method. The value of u is then updated by taking the maximum of 0 and $u+\beta\left(\mathbf{1}-\mathbf{t}-\mathbf{y}.*(\mathbf{X}^\intercal\mathbf{w})-\mathbf{y}*\mathbf{b}\right)$ in every iteration of ALM .The stopping condition for solving the subproblem is when the gradient error is less than or equal to the subtolerance or when we reach the maximum number of iterations (maxsubit), this happens when subit e quals maxsubit. The subtolerance is the tolerance set for solving our

subproblem using projected gradient descent. The stopping condition of the ALM solver is when the maximum of the pres, dres is less than or equal to the tolerance, or when the number of iterations of the ALM procedure reaches maxit. The pres is found by taking the vector constraint in our original problem of $1 - t_i - y_i * (\mathbf{w^\intercal x_i} + b))$ for all $i = 1 : n$ where n is the number of columns of X and then taking $max(0, 1 - t_i - y_i * (\mathbf{w^\intercal x_i} + b))$ for each i, and then finding the norm of that new vector. The dual residual is found by first taking the gradient of the ordinary Lagrangian function (the augmented Lagrangian function without the β term) with respect to w,b,and t. Then we add the values of the norm of gradient of b with the norm of gradient of w with respect to the ordinary Lagrangian function. Then I found the indices of vector t that are greater than zero, afterwards I took the norm of the gradient of t with respect to those indices. Then I added this value to the norm of the gradient of w added to the norm of the gradient of b. Afterwards I found all the indices of vector t that are equal to 0, and then got the new vector for the gradient of t with respect to these indices. Then I made sure that each of the values of the new gradient of t vector was less than or equal to 0 and then found the norm of this vector and added it to the norms found earlier to get the final value of the dual residual (dres).

For randata02, I performed the classification on the testing datasets, for different values of lambda and tolerance(tol). When I set $\lambda = 1000$, and opt.tol $= 1 * e^{-4}$ then I get an accuracy of 97.5%. I got the same accuracy when I set $\lambda = 700$ and opt.tol $= 1 * e^{-4}$, also when I set $\lambda = 10$ and opt.tol $= 1 * e^{-1}$, and also when I set $\lambda = 1$ and opt.tol $= 1 * e^{-1}$. When I set $\lambda = .00001$ and opt.tol $= 1 * e^{-1}$, then I got an accuracy of 86.5%. Also I set $\lambda = .0001$ and opt.tol $= 1 * e^{-2}$ and then got an accuracy of 89%. I got my best accuracy of 98% when $\lambda = .0001$, and opt.tol $= 1 * e^{-4}$ and also when $\lambda = .001$ with the same opt.tol. I got an accuracy of 87.5% when I set $\lambda = .000002$ and opt.tol $= 1 * e^{-4}$. For randata08, I performed the same procedure as I did for randata02. When I set the value of $\lambda = 500$ and opt.tol $= 1 * e^{-1}$, then I get my best accuracy of 88.5%. When the value of $\lambda = 1$ and opts.tol $= 1 * e^{-3}$, then I get an accuracy of 88%. I get the same accuracy when I set $\lambda = 10$ and opts.tol $= 1 * e^{-3}$ or opt.tol $= 1 * e^{-4}$. Also I get the same accuracy when I set $\lambda = 100$ and opts.tol $= 1 * e^{-4}$. I got an accuracy of 77.5% when I set $\lambda = .00001$ and and opts.tol $= 1 * e^{-1}$. When I set $\lambda = .0001$ and opts.tol $= 1 * e^{-1}$, then I get an accuracy of 84.5% on the testing data. Lastly, when I set $\lambda = .001$ and opts.tol $= 1 * e^{-3}$, then I get an accuracy of 84% on the testing data. below are some of the graphs

of the violation of the primal and dual feasibility

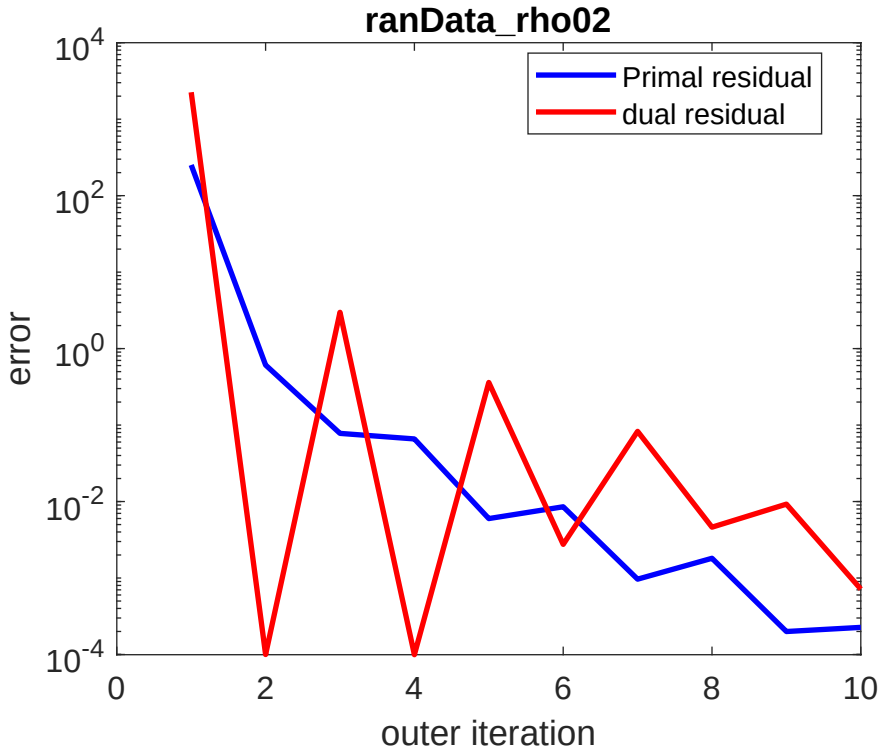

Figure 1: This is graph of violation of primal and dual feasibility

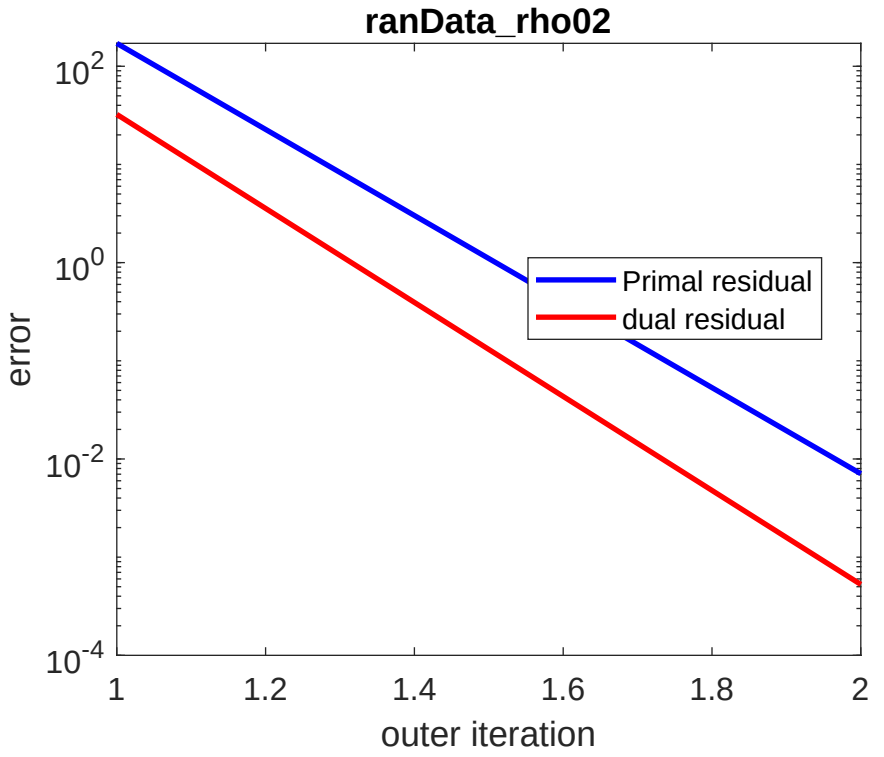

Figure 2: This is graph of violation of primal and dual feasibility

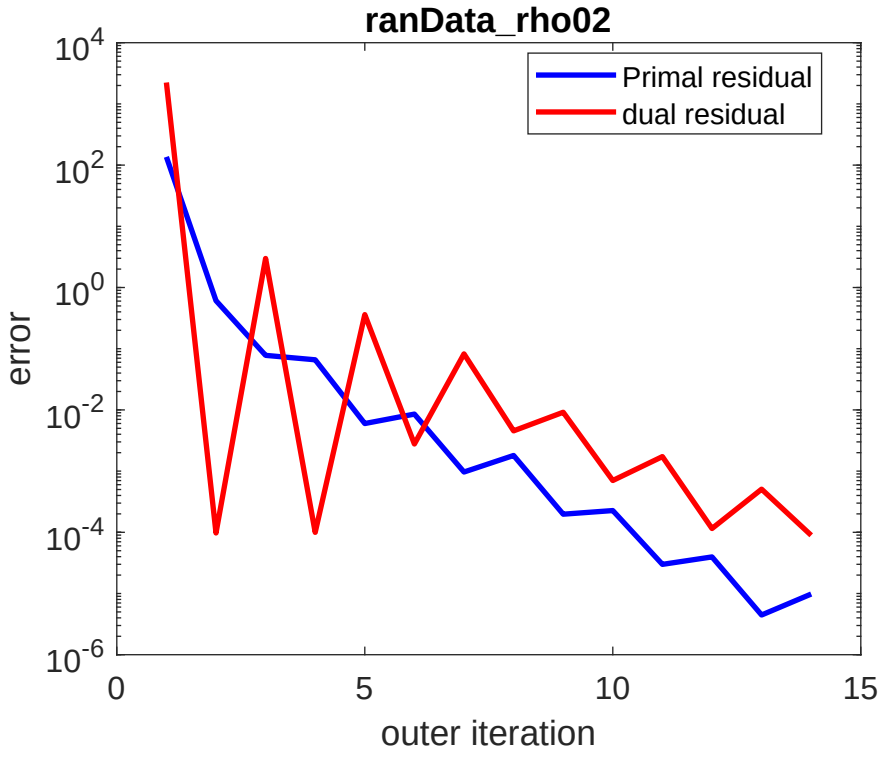

Figure 3: This is graph of violation of primal and dual feasibility

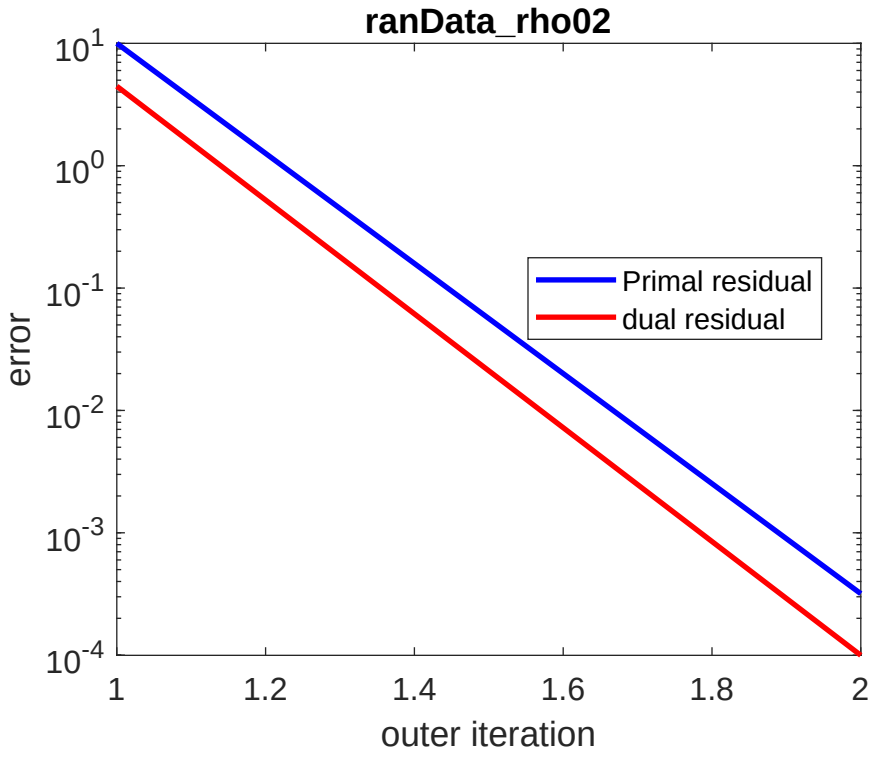

Figure 4: This is graph of violation of primal and dual feasibility

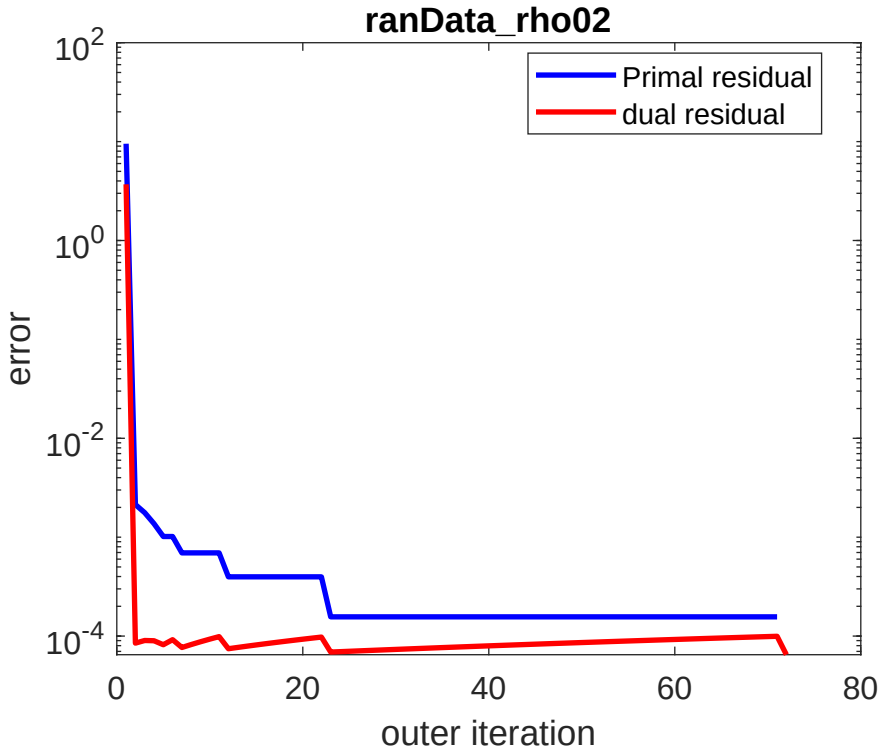

Figure 5: This is graph of violation of primal and dual feasibility

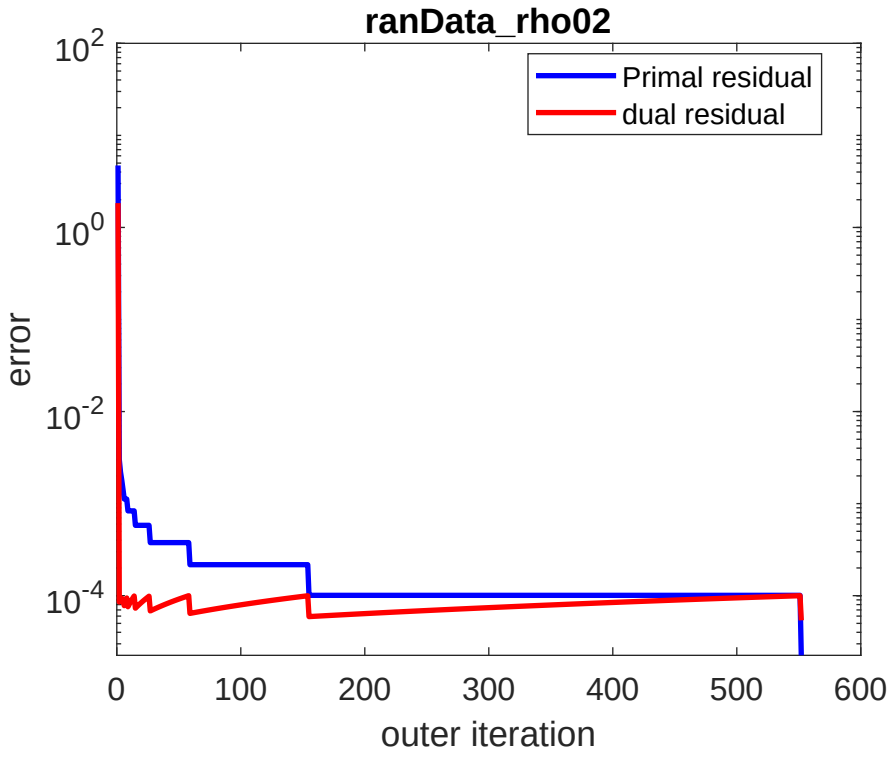

Figure 6: This is graph of violation of primal and dual feasibility

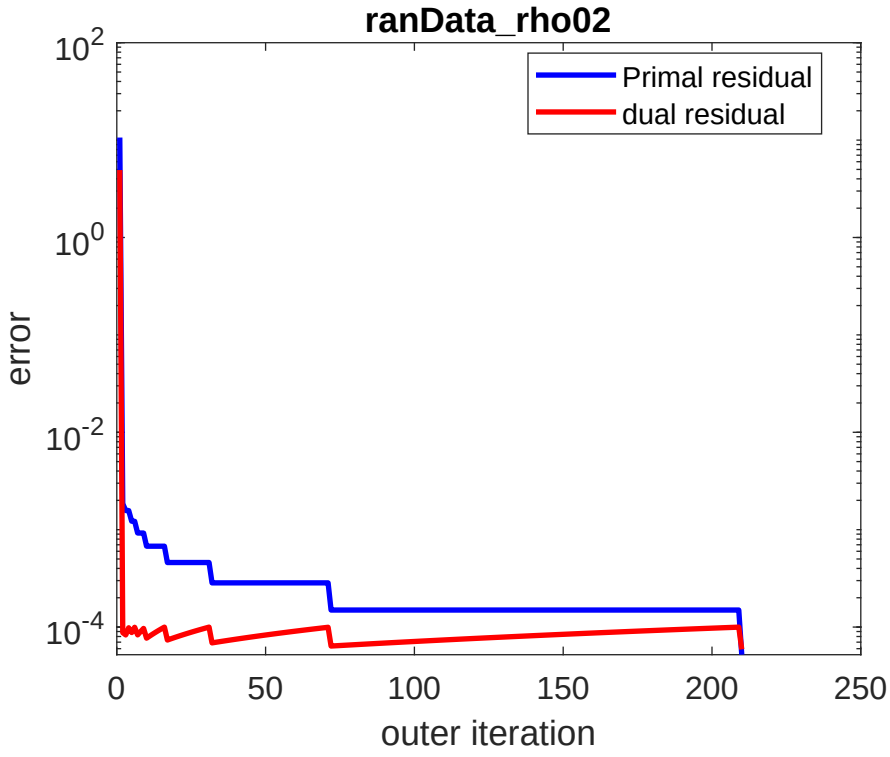

Figure 7: This is graph of violation of primal and dual feasibility

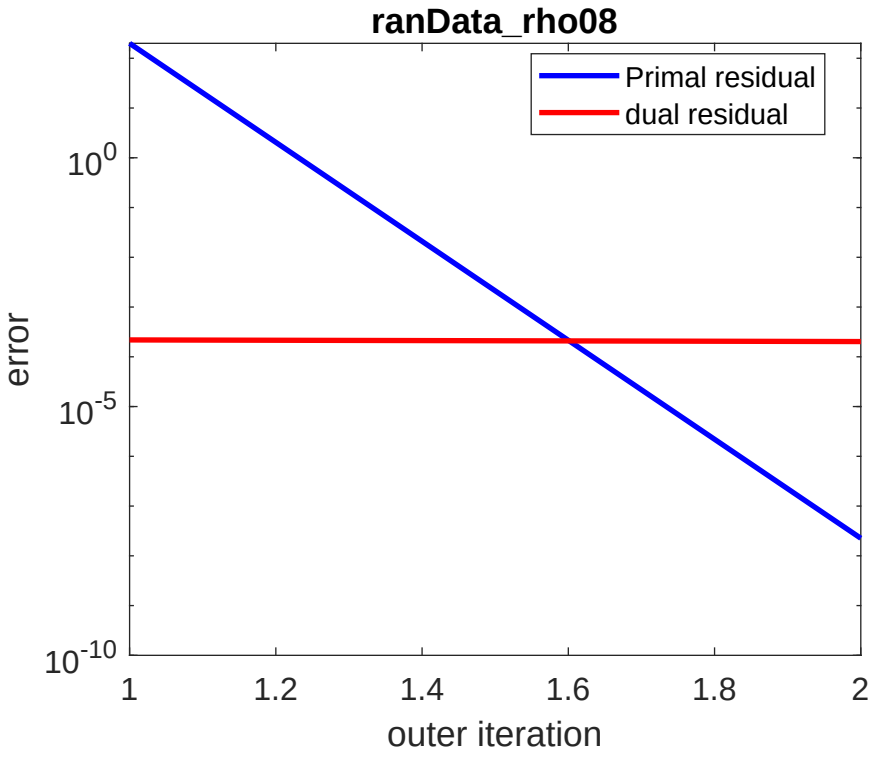

Figure 8: This is graph of violation of primal and dual feasibility

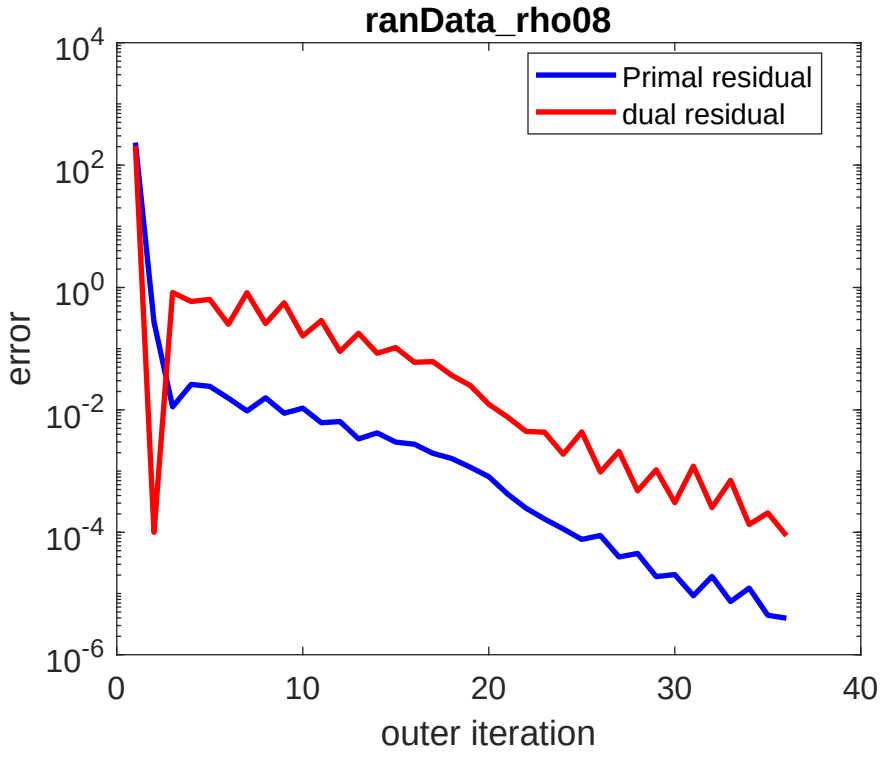

Figure 9: This is graph of violation of primal and dual feasibility

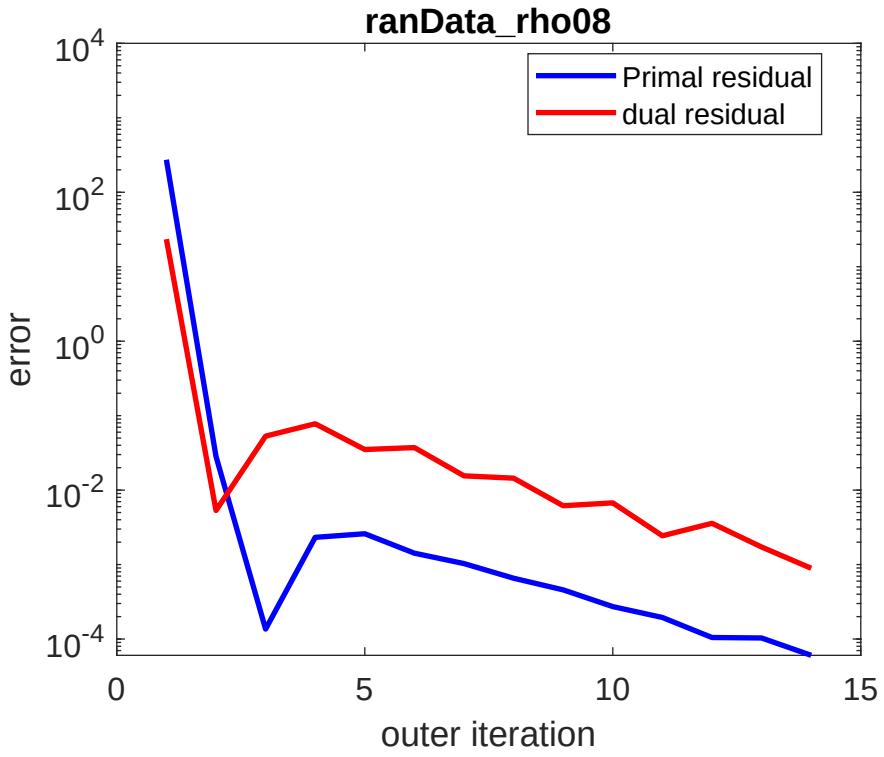

Figure 10: This is graph of violation of primal and dual feasibility

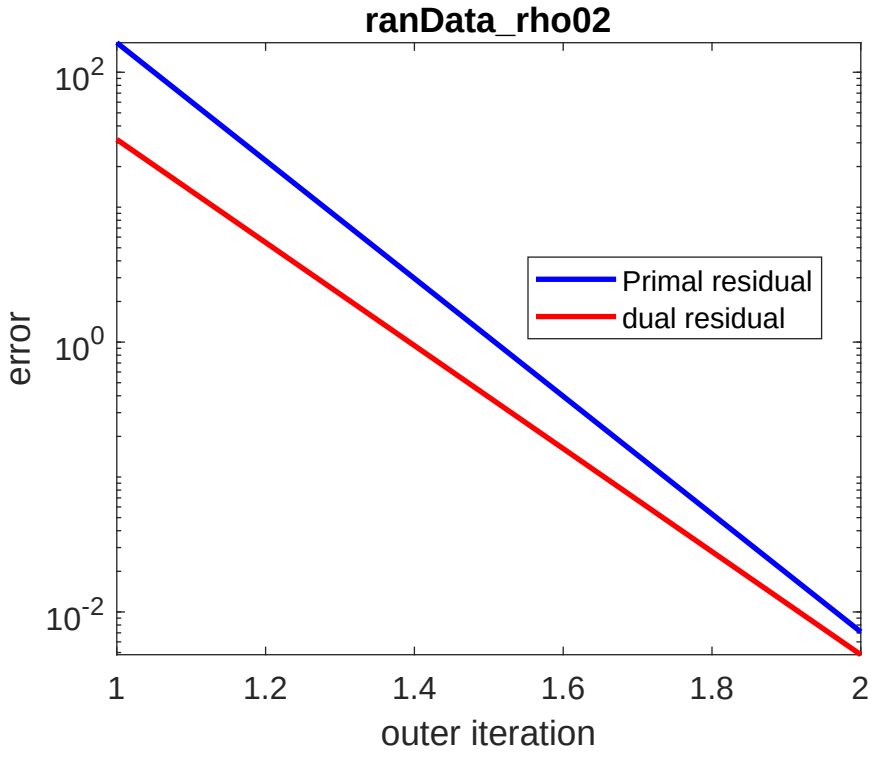

Figure 11: This is graph of violation of primal and dual feasibility

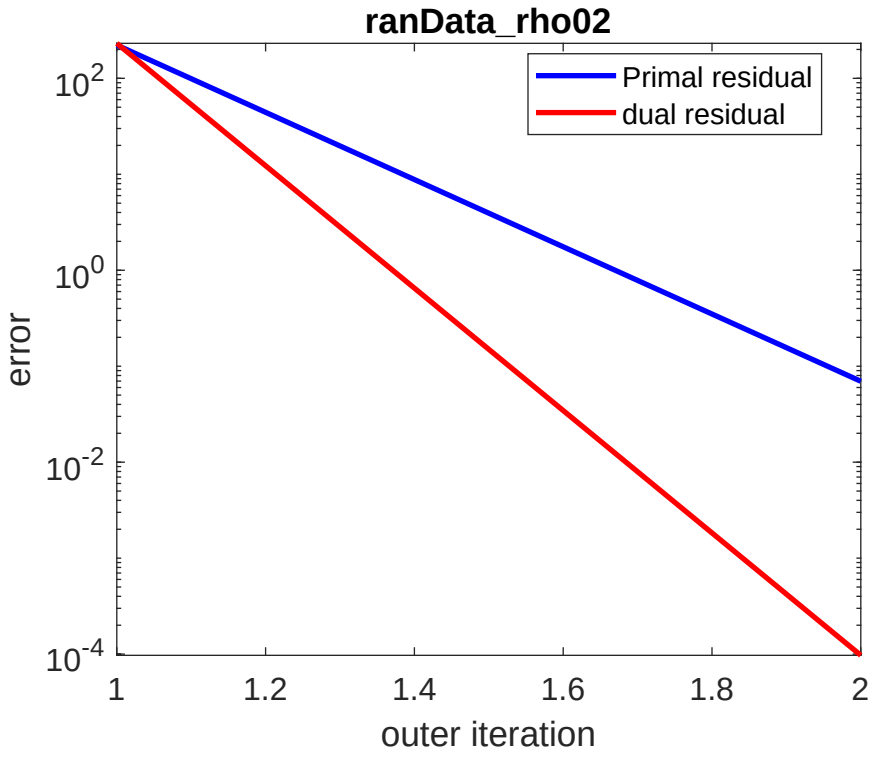

Figure 12: This is graph of violation of primal and dual feasibility

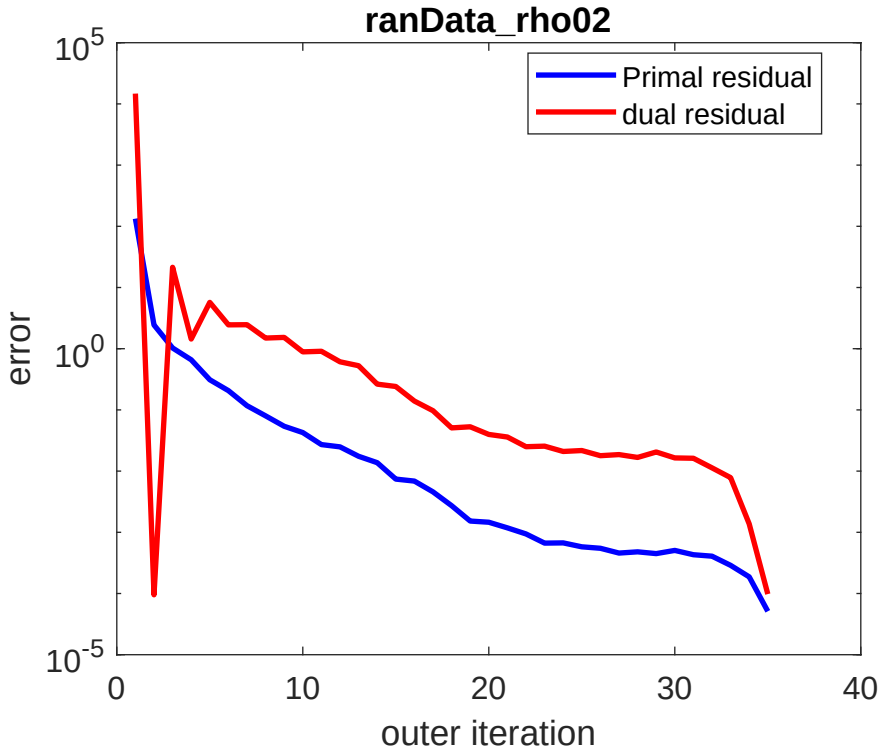

Figure 13: This is graph of violation of primal and dual feasibility

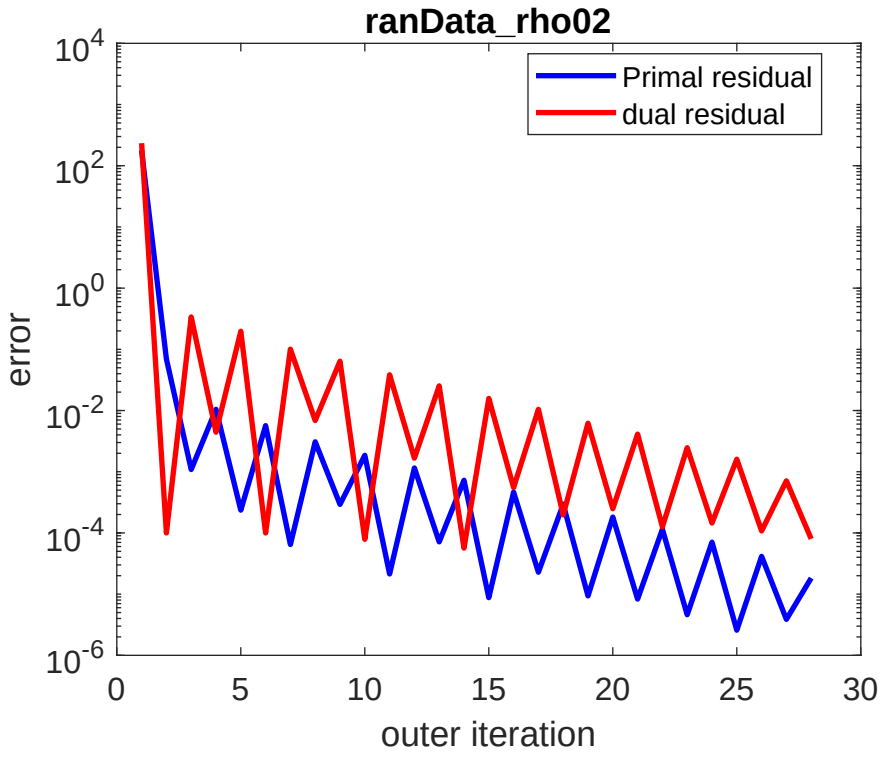

Figure 14: This is graph of violation of primal and dual feasibility

Table 1: Accuracy randata02

λ	opts.tol	Accuracy
.1	$1 * e^{-4}$	97.5%
1000	$1 * e^{-4}$	97.5%
700	$1 * e^{-4}$	97.5%
10	$1 * e^{-1}$	97.5%
1	$1 * e^{-1}$	97.5%
.0001	$1 * e^{-2}$	89%
.0001	$1 * e^{-4}$	98%
.001	$1 * e^{-4}$	98%
.000002	$1 * e^{-4}$	87.5%
.00001	$1 * e^{-1}$	86.5%

Table 2: Accuracy randata08

λ	opts.tol	Accuracy
.1	$1 * e^{-4}$	88%
1	$1 * e^{-3}$	88%
10	$1 * e^{-4}$	88%
10	$1 * e^{-3}$	88%
100	$1 * e^{-4}$	88%
.00001	$1 * e^{-1}$	77.5%
.0001	$1 * e^{-1}$	84.5%
200	$1 * e^{-4}$	87%
.001	$1 * e^{-3}$	84%
500	$1 * e^{-1}$	88.5%

Topic 2: Neyman Pearson Classification

I developed an ALM solver for topic 2 as well. The subproblem that is solved within each outer iteration is

$$\tag{4}$$

$$argmin_{w,b} L_\beta(w, b, u) \tag{5}$$

$$. \tag{6}$$

The expression $L_\beta(w, b, u)$ is the augmented Lagrangian function for this problem. The augmented Lagrangian function $L_\beta(w, b, u)$ can also be written as $\frac{1}{N^+} * \sum_{i \in N^+} log\left(1 + e^{-y_i * (\mathbf{w}^\intercal \mathbf{x_i} + b)}\right) + u * \frac{1}{N^-} * \left(\sum_{i \in N^-} log\left(1 + e^{-y_i * (\mathbf{w}^\intercal \mathbf{x_i} + b)}\right) - \alpha\right) + \frac{\beta}{2} * max\left(0, \frac{1}{N^-} * \sum_{i \in N^-} log\left(1 + e^{-y_i * (\mathbf{w}^\intercal \mathbf{x_i} + b)}\right) - \alpha\right)^2$. This equation for the augmented lagrangian function was found by first taking the function to minimize, in this case $\frac{1}{N^+} * \sum_{i \in N^+} log\left(1 + e^{-y_i * (\mathbf{w}^\intercal \mathbf{x_i} + b)}\right)$,. Then we took the inequality constraint, in this case $\frac{1}{N^-} * \left(\sum_{i \in N^-} log\left(1 + e^{-y_i * (\mathbf{w}^\intercal \mathbf{x_i} + b)}\right)\right) \leq \alpha$.

I wanted to get the inequality constraint into the form $g_i(w,b) \leq 0$ where $g_i(w,b)$ represents the new inequality in this case, $g_i(w,b) = \sum_{i \in N^-} log\left(1 + e^{-y_i*(\mathbf{w^T x_i}+b)}\right) - \alpha$, since this inequality is ≤ 0 . I then plugged in this inequality into the formula for the augmented lagrangian function with quadratic constraint to get the augmented lagrangian function $L_\beta(w,b,u)$. Since the value of $g_i(w,b)$ is a scalar and not a vector then the value of our multiplier u must be a scalar and not a vector in this case. The value of N^+ in the minimization function was found by counting the number of values in the vector y that are equal to 1. The value of N^- in the constraint was found by counting the number of values in the vector y that are equal to -1. This was found by first setting the sign(y) into a variable in matlab. Since 1 is positive and -1 is negative and those are the only two values of y, I decided to separate y by its sign in order to count the number of positive and negative values in y. Then in order to find the total number of positive and negative values in y, I utilized the sum function based on sign(y) computed earlier. I solved the subproblem utilizing gradient descent. First I set $w = w0$ where $w0$ is the initial value of w and $b = b0$ where $b0$ is the initial value of b computed the initial value of the gradient of the augmented Lagrangian function. When I take the gradient with respect to the objective function part of the augmented lagrangian $\frac{1}{N^+} * \sum_{i \in N^+} log\left(1 + e^{-y_i*(\mathbf{w^T x_i}+b)}\right)$ I make sure to only consider the positive part of X and y as a part of the gradient. The positive part of X is found by taking the columns of X whose corresponding labels of y are positive. For example in the training data set of ranDatarho02, the first fifty entries of y are positive, thus the first fifty columns of X are the positive parts of X. The positive part of y is found by taking the entries of vector y equal to one. When I take the gradient with respect to the constraints of the augmented lagrangian function $u * \frac{1}{N^-} * \left(\sum_{i \in N^-} log\left(1 + e^{-y_i*(\mathbf{w^T x_i}+b)}\right) - \alpha\right) + \frac{\beta}{2} * max\left(0, \frac{1}{N^-} * \sum_{i \in N^-} log\left(1 + e^{-y_i*(\mathbf{w^T x_i}+b)}\right) - \alpha\right)^2$, then I only consider the negative part of X and the negative part of y. The negative part of X is found by taking the columns of X whose corresponding label of y is negative. For example, in the training dataset for ranDatarho02, the entries fifty one through two hundred of y are negative, thus the columns fifty one through two hundred of X are the negative parts of X. The negative part of y is found by getting a vector that contains the entries of y equal to -1. Similar to the last problem I vectorized the augmented lagrangian function in a similar way using the exp function in matlab.In order to vectorize the first part of the augmented lagrangian function $\frac{1}{N^+} * \sum_{i \in N^+} log\left(1 + e^{-y_i*(\mathbf{w^T x_i}+b)}\right)$; I first wanted to create a column vector where each row has components of this function such that the first row of this column vector has the entry $log\left(1 + e^{-y_1*(\mathbf{w^T x_1}+b)}\right)$, the second row of this column vector has the entry $log\left(1 + e^{-y_2*(\mathbf{w^T x_2}+b)}\right)$, ..., up until the mth row of the column vector has the entry $log\left(1 + e^{-y_m*(\mathbf{w^T x_2}+b)}\right)$ representing only the columns of x in xpos in this case and only ypos in this case as well. The expression

$w^\intercal \mathbf{Xpos}$ gives a row vector in which the first column is $w^\intercal \mathbf{x_1}$ where x_1 is first column of the positive part of X, the second column is $w^\intercal \mathbf{x_2}$ where x_2 is second column of the positive part of X,...., up until the mth column which is $w^\intercal \mathbf{x_m}$ where x_m is the mth column of the positive part of X. I took the transpose of $w^\intercal \mathbf{X}$ which is equivalent to $X^\intercal \mathbf{w}$ in order to get a column vector whose first row is $w^\intercal \mathbf{x_1}$, second row is $w^\intercal \mathbf{x_2}$, ..., mth row is $w^\intercal \mathbf{x_m}$ and then I multiplied this vector componentwise by -y . Then I multiped negative of the pos part of y times b, and then put this whole expression into the exp function in matlab. The expression then became $log(1 + exp((-ypos. * (transpose(xpos) * w)) - (ypos * b)))$. In order to get the summation of this vector I multiplied this vector by a column vector of ones that has the same number of rows as columns of X positive and then I divided by the number of positive values of y or N^-. I vectorized the other two summations of the augmented lagrangian function in a similar manner except using the negative part of X and the negative part of y , and the column vector of ones that has the same number of rows as columns of the negative part of X and then I divided by the number of negative values of y or N^-. The first part of the function The iterative update formula in gradient descent for w is found by taking w0(initial value of w) and then subtracting the stepsize alpha2 multiplied by the gradient of the augmented Lagrangian function with respect to w0 from w, w = w0 - (alpha2 * gradient of augmented Lagrangian with respect to w0). The iterative update formula in gradient descent for b is found by taking b0(initial value of b) and then subtracting the stepsize alpha2 multiplied by the gradient of the augmented Lagrangian function with respect to b0 from b, b = b0 - (alpha2 * gradient of augmented Lagrangian with respect to b0). I also update the iteration number subit by one for the projected gradient method, for each iteration of the projected gradient method. I also update the value of iter or the number of iterations of our alm method by one through each iteration of our alm method. The value of our multiplier u is also updated through each iteration of the alm method. The value of u is first initialized to 0 in our alm method. The value of u is found by taking the maximum of 0 and $u + \beta * g_i(w, b)$. Thus, $u = max(0, u + \beta * g_i(w, b))$. The gradient of our augmented lagrangian function with respect to w,and b is updated after every iteration in our ALM solver.The stopping condition for our subproblem occurs when the gradient error is less than or equal to the subtolerance or when the iteration number of our subproblem (subit) equals maxsubit or the maximum number of iterations. The subtolerance is the tolerance set for solving our subproblem using gradient descent. The gradient error is found by first taking the gradient of the augmented Lagrangian function with respect to w and b. Then we find the norm of the gradient of w and the norm of the gradient of b with respect to the augmented lagrangian function and add those values together to get the gradient error. The stopping condition for our ALM solver occurs when the maximum of the primal and dual residual is less than or equal to the tolerance set for our ALM method, or when the number of iterations of the ALM method (iter) equals maxit or the maximum number of iterations of the ALM method. The primal residual is found by taking the constraint $g_i(w, b)$ and then finding the maximum

of 0 and $g_i(w,b)$, $max(0, g_i(w,b))$. If the constraint is greater than 0 than we have a primal residual, otherwise the primal residual is 0. After we take the maximum of 0 and $g_i(w,b)$, I then found the absolute value of this new expression to get the value for our primal residual. In order to find the dual residual, I first took the gradient of the ordinary Lagrangian function,also known as the augmented Lagrangian function without a β term with respect to w and b. Then I added the norm of the gradient of w with the norm of the gradient of b with respect to the ordinary lagrangian function together in order to get the value for the dual residual. The initial value of the primal and dual residual are the first items computed in our alm solver. The primal and dual residual are recomputed in every iteration of our ALM solver based on the new values of w and b.

The false negative error is the percentage of data points that are classified as negative, but should actually be positive. In order to find the false negative error I did one hundred minus the sensitivity. Sensitivity corresponds to the percentage of positive datapoints that are correctly classified as positive. This corresponds with the accuracy on the positive dataset. Thus I did one hundred minus the accuracy on the positive dataset to get the false negative error. This corresponds with percentage of data points that are classified as negative, but should actually be positive. The false positive error is the percentage of datapoints that are classified as positive but should be negative. This was found by taking one hundred minus the percentage of negative datapoints that are correctly classified as negative. This corresponds with the accuracy on the negative dataset. Thus I did one hundred minus the accuracy on the negative dataset to get the false positive error. For the randata02 dataset, I tested the false positive and false negatives for various values of α. The default value of $\alpha = .1$ gives me a false positive error of 7% and a false negative error of 13%.When I set $\alpha = .004$, then I got a false negative error of 10% and a false positive error of 7%. When I set $\alpha = .0005$ then I got a false negative error of 7% and a false positive error of 4%. The next value of α I picked was .00001 and when I did this value of α I got a false negative error of 10% and a false positive error of 7%. Afterwards I picked the next value of α to be .01 and I got a false negative error of 11% and a false positive error of 7%. When I set $\alpha = 1000$ I got a false positive error of 12% and a false negative error of 15%. I also tested various values of α for the randata08 dataset as well. When I set alpha to the default value of .1, I get a false positive error of 21% and a false negative error of 21% as well. When I set $\alpha = 100$ then I get a false positive error of 28% and a false negative error of 28% as well. The next value I set for α was 14, when $\alpha = 14$, I get a false positive error of 21% and a false negative error of 26%. When $\alpha = 170$, I get a false positive error of 24% and a false negative error of 27%. Afterwards I set $\alpha = .0009$ and I got a false positive error of 20% and a false negative error of 27%. Then I set $\alpha = .005$ and I got a false positive error of 22% and a false negative error of 24%. Later I set $\alpha = .0008$ and I got a false positive error of 23% and a false negative error of 22%. Lastly I set $\alpha = .002$ and I got I got a false positive error of 26% and a false negative error of 15%.

Table 3: Accuracy randata02 2nd problem

α	false positive error	false negative error
.1	7%	13%
.004	10%	7%
.0005	7%	4%
.00001	7%	10%
.01	7%	11%
1000	12%	15%

Table 4: Accuracy randata08 2nd problem

α	false positive error	false negative error
.1	21%	21%
100	28%	28%
14	21%	26%
170	24%	27%
.0009	20%	27%
.005	22%	24%
.0008	23%	22%
.002	15%	26%

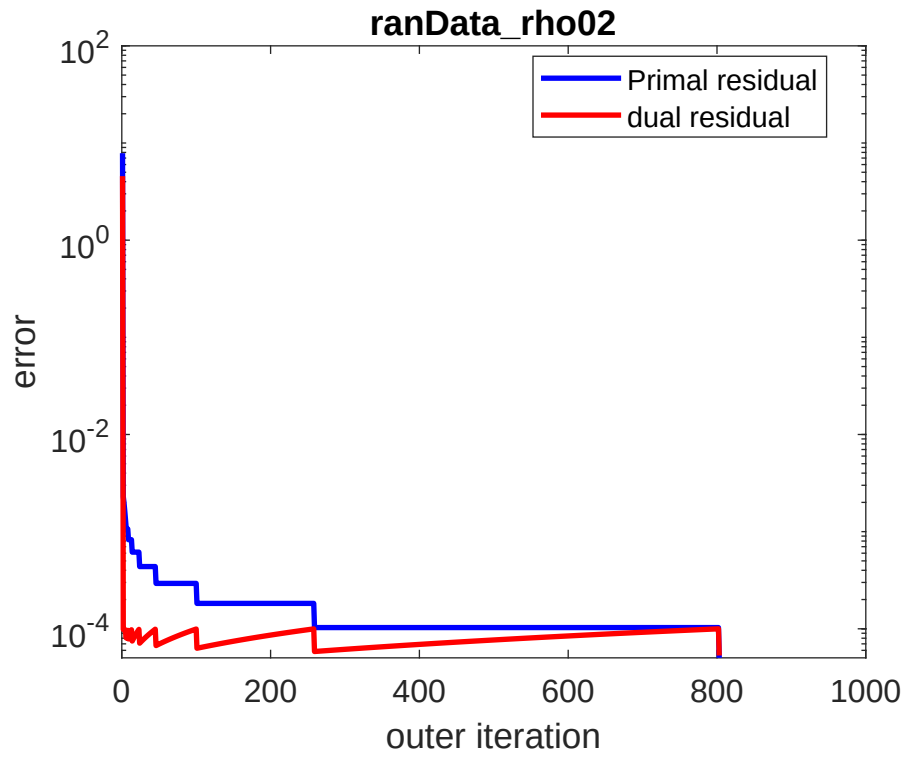

Figure 15: This is graph of violation of primal and dual feasibility for lambda = 100 and opts.tol = 1e-4

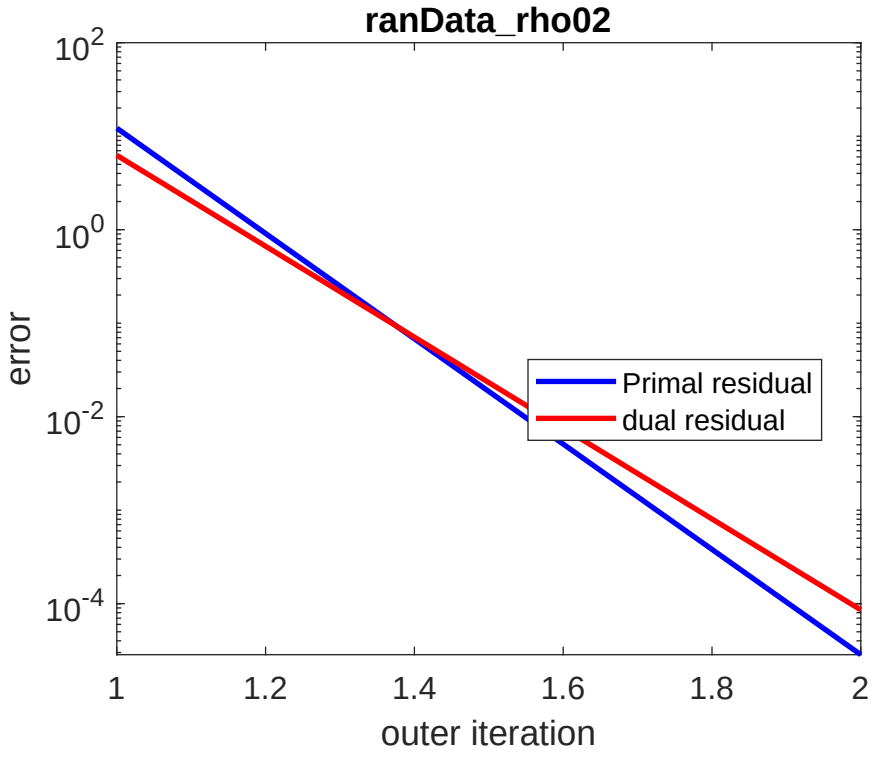

Figure 16: This is graph of violation of primal and dual feasibility

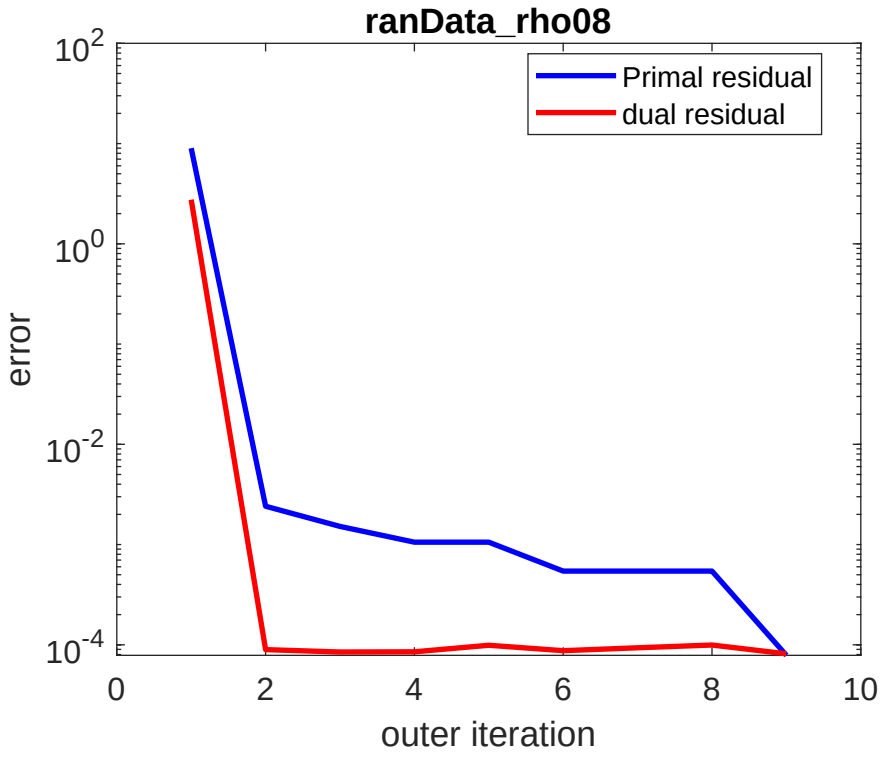

Figure 17: This is graph of violation of primal and dual feasibility