QP Solvers and their Applications

Mode: Application

1st Aditya Vats
Mechanical Engineering. IITK
Roll Number: 19807069

2nd Pratikhya Ranjit
Electrical Engineering. IITK
Roll Number: 19807639

***Abstract*—This term paper reviews three approaches to solving a Convex Quadratic Program - Active Set Method, ADMM, and Interior Point Method. We summarize the algorithms and illustrate details on their implementations. We also compare them on the basis of their convergence rate, accuracy, and complexity. We pick two applications that are used frequently in the domains of ML and robotics - Support Vector Machines (SVM) and Model Predictive Control (MPC). The theory behind both the applications is also described.**

***Index Terms*—Quadratic Program, Primal, Dual, KKT Conditions**

I. Introduction

Quadratic Programs are a standard formulation of a class of minimization problems. The objective is formulated as a quadratic function over the optimization variable. A typical quadratic program looks as follows:

$$\begin{aligned} \min_{x} \quad & J \triangleq \frac{1}{2}x^T Q x + c^T x \\ \text{s.t.} \quad & Ax = b \\ & Gx \leq h \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the optimizing variable. The objective function is made by $Q \in S^n_{++}$ and $c \in \mathbb{R}^n$. The equality constraints are defined by $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The inequality constraints are defined by $G \in \mathbb{R}^{p \times n}$ and $h \in \mathbb{R}^p$. The above problem is convex if $Q \in S^n_{++}$ in which case, the global solution is guaranteed to be found.
This paper reviews various solvers for the general convex quadratic problem, and the solvers are compared on their convergence rate, their accuracy. Amongst various solvers for quadratic programs [1], the following three solvers are compared.

1) Dual Active-Set Method (ASM)
2) ADMM
3) Interior Point Method (IPM) (using qpSwift package)

Note 1: Only Active Set Method and ADMM are implemented in this paper. IPM is used as a baseline solver and qpSWIFT package is used for IPM solver. [2]

Note 2: In all three algorithms, sparsity in coefficient matrices is exploited using Cholesky decomposition (LDL^T) and sparse direct methods are used to solver matrix equalities. The approach is originally used in [2] but implemented by us for ADMM and ASM methods as well.

II. Dual Active Set Algorithm

Though there are different flavors in the active set class of solvers [3] [4], leveraging dual or primal form of objective, we'll focus on minimizing the dual objective in this paper. The algorithm we are implementing follows from the one provided in [3]

A. Intuition

The primal objective is converted to its dual problem:

$$\min_{\lambda \geq 0} \quad J_d \triangleq \frac{1}{2}\lambda^T H H^T \lambda + c^T \lambda \tag{2}$$

where the new variables are

$$H \triangleq AR^{-1}, \quad v \triangle R^{-T} f, \quad d \triangleq Hv, \tag{3}$$

and R is an upper triangular Cholesky factor of Q i.e. $Q = R^T R$.

The steps in the algorithm are:

1) A working set W is a set of constraints that are 'active', i.e. the dual variable λ of this set of constraints are free to vary. In the dual formulation, accordingly, H would be replaced by H_w which is a matrix with the rows of H corresponding to the working set. λ is similarly replaced with λ_w.
2) The problem is then solved to obtain λ^*_w. The dual feasibility is checked through $\lambda^*_w \geq 0$. The primal feasibility for all the constraints not in working set W is checked by calculating the primal slack μ. If $\mu \geq 0$, then optimal solution is obtained.
3) In case the feasibility is violated, that constraint is added to the working set W and steps 1 and 2 are repeated again. This is done iteratively till all the constraints are satisfied i.e. working set W contains all the constraints.

Note 3: The dual active-set algorithm requires $Q \succ 0$. In order to handle cases where $Q \succeq 0$, we apply proximal point iterations as stated in [3]. The steps in proximal point iterations include:

1) In the primal problem (1), replace Q with $Q + \epsilon I$. Compute R_ϵ as the upper triangular Cholesky factor of Q_ϵ. Compute H_ϵ using (3). Initialise x, λ, W.
2) Compute $v = R_\epsilon^{-T}(f - \epsilon x)$ and $d = b + M_\epsilon v$.
3) Compute new values of x, λ, W by applying dual active-set algorithm with values computed in Step 1 and 2.
4) Repeat Step 2 and 3 until convergence.

Algorithm 1 Dual Active-Set Algorithm

Input: $H, c, v, R^{-1}, W_0, \lambda_0$
Output: x^*

while Not Converged **do**
  if $H_w H_w^T$ is nonsingular **then**
    $[\lambda_i^*]_{W_i} \leftarrow solve(H_w H_w^T [\lambda_k^*]_{W_i} + c_i = 0)$
    if $\lambda_i^* \geq 0$ **then**
      $[\mu_i]_{\bar{W}_i} \leftarrow \mathrm{H}_{\bar{w}} H_w^T [\lambda_i^*]_{W_i} + \bar{c_i}$
      $\lambda_{i+1} \leftarrow \lambda_i^*$
      if $\mu_i \geq 0$ **then** Optimum found. Go To Line
      else
        $j \leftarrow argmin_{r \in \bar{W}_i} [\mu_k]_r$
        $W_{i+1} \leftarrow W_i \cup j$
      end if
    else
      $p_i \leftarrow \lambda_i^* - \lambda_i$
      $B \leftarrow \{j \in W_i : [\lambda_i^*]_j < 0\}$
      $[\lambda_{i+1}, W_{i+1}] \leftarrow FIX\_VAR(\lambda_i, W_i, B, p_k)$
    end if
  else
    $[p_i]_{W_i} \leftarrow solve(H_w H_w^T [p_i]_{W_i} = 0, p_i^T c < 0)$
    $B \leftarrow \{j \in W_i : [p_i]_j < 0\}$
    $[\lambda_{i+1}, W_{i+1}] \leftarrow FIX\_VAR(\lambda_i, W_i, B, p_k)$
  end if
  $i \leftarrow i + 1$
end while
Return $x^* \leftarrow -R^{-1}(H_i^T [\lambda_k^*]_{W_i} + v)$

III. ADMM

A. Intuition

Alternating Direction Method of Multipliers (ADMM) is an alternating optimization algorithm of solving quadratic problems [5]. It breaks the problem into subproblems and solves them iteratively until it converges to the global optimum. At each iteration, a fixed number of subproblems are solved which consist of minimizing the Lagrangian wrt to one variable at-a-time by fixing all the other variables to their values in the previous iteration. The number of subproblems depends on the number of variables including auxiliary variables present in the optimization problem.

B. Algorithm

The equality and inequality constraints can be represented in a combined form as

$$l <= Ax <= u \tag{4}$$

where l is the lower bound and u is the upper bound of each inequality constraint. Equality constraints are the constraints where $l = u$.
In the algorithm, optimization variables x_0, y_0, z_0 are initialized as 0. The parameters σ, ρ determine the step size to be used to update a variable in each iteration and α is defined as the relaxation parameter. Their values are based on that stated in OSQP [5] and have been found to produce desired results. The algorithm terminates when the primal and dual residuals are below their respective tolerances.

Algorithm 2 ADMM Algorithm

Initialise x_0, y_0, z_0 and $\rho > 0, \sigma = 10^{-6}, \alpha = 1.6, k = 0$
while not converged **do**

$$\begin{bmatrix} Q + \sigma I & A^T \\ A & -\rho^{-1} I \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ \tilde{\nu}^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - c \\ z^k - \rho^{-1} y^k \end{bmatrix}$$

$\tilde{z}^{k+1} = z^k + \sigma^{-1}(\nu^{k+1} - y^k)$
$x^{k+1} = \alpha \tilde{x}^{k+1} + (1 - \alpha) x^k$
$z^{k+1} = projection(\alpha \tilde{z}^{k+1} + (1 - \alpha) z^k + \rho^{-1} y^k)$
$y^{k+1} = y^k + \rho(\alpha \tilde{z}^{k+1} + (1 - \alpha) z^k - z^{k+1})$
end while

In order to achieve faster convergence, the original problem is scaled down to a smaller problem using a modified version of Ruiz equilibration as stated in the OSQP [5]. In this method, a scaling matrix S and parameter d is computed. The form of S is

$$S = \begin{bmatrix} D & \\ & E \end{bmatrix} \tag{5}$$

and the new problem becomes

$$\begin{aligned} minimize \quad & \frac{1}{2} \tilde{x}^T Q \tilde{x} + \tilde{c}^T \tilde{x} \\ s.t. \quad & \tilde{l} <= \tilde{A} \tilde{x} <= \tilde{u} \end{aligned}$$

with $\tilde{x} = D^{-1}x, \tilde{Q} = dDQD, \tilde{c} = dDc, \tilde{A} = EAD, \tilde{l} = El$ and $\tilde{u} = Eu$.

We did not implement the solution polishing method used in OSQP [5]. Yet, we have achieved high accuracy and fast convergence comparable to the original work in OSQP.

IV. Interior Point Method

This method was not implemented by us and the software qpSWIFT was used for IPM Solvers [2]

A. Intuition

IPM is used to solve quadratic problems with inequality constraints by reducing them to equality constraints. The following steps are involved in IPM:

1) The inequalities are replaced with log barrier penalty function and the Lagrangian is formed.
2) The first order optimality conditions (i.e the KKT conditions) are formulated as a matrix equation.
3) The Newton method is applied (i.e. derivative of the above matrix equation is obtained). The solution to the differentiated matrix system gives the search directions for optimization variable, say x.
4) The variable x is changed by some constant(α) $\times$ search direction(δx). So $x = x + \delta x * \alpha$
5) Step 3 and 4 are repeated till convergence.

Using slack variable s, the primal problem (1) can be written as

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^TQx + c^Tx \\ \text{s.t.} \quad & Ax = b \\ & Gx + s = h \\ & s \geq 0 \end{aligned} \tag{6}$$

The Lagrangian with μ as barrier parameter is given by

$$\begin{aligned} L_\mu(x,y,z) = & \frac{1}{2}x^TQx + c^Tx + y^T(Ax-b) + \\ & z^T(Gx - h + s) - \mu\sum_i s_i \end{aligned} \tag{7}$$

The KKT conditions derived from (7), also called the central path equations, are solved to reach the optimal point.

B. Algorithm

The main steps involved in solving the central path equations are given below.

1) Initialize (x_0, y_0, z_0, s_0) by solving

$$\begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -I \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ \hat{z} \end{bmatrix} = \begin{bmatrix} -c \\ b \\ h \end{bmatrix} \tag{8}$$

$$s_0 = \begin{cases} -\hat{z} & \alpha_p < 0 \\ -\hat{z} + (1+\alpha_p)1 & o/w \end{cases}$$

$$\text{where} \quad \alpha_p = inf\{\alpha | -\hat{z} + \alpha 1 \geq 0\}$$

$$z_0 = \begin{cases} \hat{z} & \alpha_d < 0 \\ \hat{z} + (1+\alpha_d)1 & o/w \end{cases}$$

$$\text{where} \quad \alpha_d = inf\{\alpha | \hat{z} + \alpha 1 \geq 0\}$$

2) Compute residuals of the central path (r_x, r_y, r_z) as

$$\begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -W^TW \end{bmatrix} \begin{bmatrix} x^k \\ y^k \\ z^k \end{bmatrix} + \begin{bmatrix} c \\ -b \\ s^k - h \end{bmatrix} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \tag{9}$$

3) Compute affine direction $(\delta x_a, \delta y_a, \delta z_a, \delta s_a)$ by solving

$$\begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -W^TW \end{bmatrix} \begin{bmatrix} \delta x_a \\ \delta y_a \\ \delta z_a \end{bmatrix} = \begin{bmatrix} -r_x \\ -r_y \\ -r_z - W^T(r_s \diamond \lambda) \end{bmatrix}$$

$$\delta s_a = -W^TW\delta z + W^T(r_s \diamond \lambda) \text{ with } r_S = -\lambda \circ \lambda \tag{10}$$

where W is the Nesterov-Todd scaling matrix [9] and symbols $\diamond, \circ$ are based on qpSWIFT [2].

4) Compute parameters α, σ, ρ as

$$\begin{aligned} \alpha &= sup\{\alpha \in [0,1] | (s,z) + \alpha(\delta s_a, \delta z_a) \geq 0\} \\ \rho &= \frac{(s + \alpha\delta s_a)^T(z + \alpha z_a)}{s^Tz} \\ \sigma &= max\{0, min\{1, \rho\}^3\} \end{aligned} \tag{11}$$

5) Compute the search direction $(\delta x_f, \delta y_f, \delta z_f, \delta s_f)$ using (10) with

$$r_s = -\lambda \circ \lambda - (W^{-T}\delta s_a) \circ (W\delta z_a) + \sigma\mu 1 \tag{12}$$

and $X = X + \alpha\delta X$

V. Application

A. Support Vector Machine

The dual problem of linear SVM [6] which allows some misclassified points (Soft-Margin SVM) is given by

$$\begin{aligned} \min_{a,w} \quad & \frac{1}{2}a^Ta + \lambda\sum_i w_i \\ s.t \quad & y_i(a^Tx_i) + 1 <= w_i \\ & 0 <= w_i \end{aligned} \tag{13}$$

where a is the weight vector, w_i is the distance from margin which is non-zero for misclassified points and $\{x_i, y_i\}$ are the datapoints.

In matrix form, the problem can be re-written as

$$\begin{aligned} min_{a,w} \quad & \frac{1}{2}a^Ta + \lambda 1^Tw \\ s.t \quad & diag(y)Ax + 1 <= w \\ & 0 <= w \end{aligned} \tag{14}$$

where $w = [w_1, w_2, ..., w_n]$ and $diag(y)$ is a diagonal matrix with diagonal entries of y_i. This form is solved by the QP solvers.

We choose to solve a binary classification problem by using SVM described above. We pick two random classes from the IRIS dataset [10] and create a binary classification dataset. We solve the QP problem (14) using each solver on the training dataset which gives an estimate of the linear discriminator. We evaluate the solvers on the test dataset.

B. Model Predictive Control

The control problem chosen to compare the QP solvers is the control of an inverted cart-pendulum [7]. Fig. 1 illustrates the setting of the problem. A pre-calculated trajectory is defined over the position of the cart and the angle θ of the pendulum. The control problem is formulated as:

$$\begin{aligned} \min_{\dot{X}_d} \quad & \sum_{k=1}^{h} ||X_r[t_i + k\delta t] - (X_i + \dot{X}_d k\Delta t)||_2^2 \\ \text{s.t.} \quad & LB \leq \dot{X}_d \leq UB \end{aligned} \tag{15}$$

where X_r is the reference trajectory, $X_i = (x, \dot{x}, \theta, \dot{\theta})$ is the state of the pendulum at time-step i. Since, it is an MPC problem, the residual is summed over a given horizon (chosen by the user). The control input computed is the desired acceleration or $\dot{X}_d$.

There are two variations relating the application of this control input $\dot{X}_d$:

- **Kinematic**: $\dot{X}_d$ used to propagate X using Forward Euler integration
- **Dynamic**: Force F on cart is calculated by calling Inverse Dynamics on control input $\dot{X}$, then the state is calculated using forward dynamics, with input being F. Forward dynamics implies subjecting the model to Force F and propagating its state. Inverse Dynamics is used to get the F corresponding to the desired $\dot{X}_d$

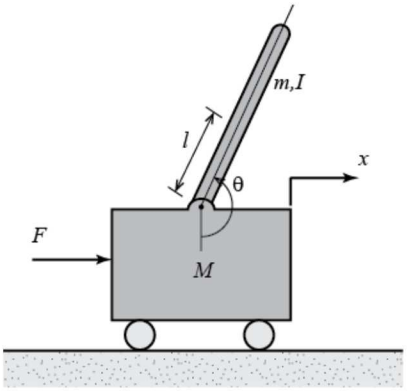

Fig. 1. Cart-Pendulum Problem. The states are θ and x

For the sake of completeness, following are the equations of motions of the above system [8]:

$$\begin{aligned}(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}cos(\theta) - ml\dot{(\theta)}^2 sin(\theta) = F \\ (I+ml^2)\ddot{\theta} + mglsin(\theta) + ml\ddot{x}cos(\theta)\end{aligned} \tag{16}$$

VI. Results

We compare the performance of each solver based on their convergence rate (total number iterations required for convergence), time taken to run one iteration by the solver, total time taken by the solver to converge to optimal solution and accuracy of the solver. We benchmark our implemented solvers ADMM and Active-Set methods with the qpSWIFT [2] which is based on IPM. We have chosen qpSWIFT as benchmark because of its highly optimized approach.

A. Accuracy

The accuracy metric for each application is different and is computed as follows:

1) SVM: Since it is a binary classification problem, we use

$$\text{Accuracy} = \frac{\text{No of correct labels predicted}}{\text{Total no of labels predicted}}$$

2) MPC: The metric used here is the MSE between the reference trajectory and the actual trajectory.

$$\text{Score} : (\hat{x}, \hat{\theta}) = \sqrt{\sum (X_r(t) - X(t))^2}$$

where $X_r(t)$ is the reference trajectory and $X(t)$ is the actual trajectory. $\hat{x}, \hat{\theta}$ define the score over variables x and θ respectively. Lower score implies better accuracy of the solver. Among ADMM and ASM, the solver that has score closer to IPM is better.

B. Convergence Rate

It is computed as the total number of iterations required for the solver to converge to the optimal solution.

1) SVM: The linear discriminator is estimated by solving one QP problem in the entire training process. The convergence rate is equal to the total number of iterations required to converge.

2) MPC: The trajectory is estimated over 100 time steps and in each time step a QP problem is solved. The number of iterations to converge in each time step is stored and convergence rate is computed as the mean number of iterations to converge over 100 time steps.

TABLE I
SUPPORT VECTOR MACHINE

Solver	Performance			
	Convergence Rate	*Total time(s)*	*Time per iter(ms)*	*Accuracy*
ADMM	549	0.083	0.15	1.0
Active-Set	6	0.099	16.5	1.0
IPM	9	0.26m	0.03	1.0

* Accuracy = $\frac{\#\text{correct label predictions}}{\#\text{total label predictions}}$ *iter Iteration

TABLE II
MODEL PREDICITVE CONTROL

Solver	Performance				
	CR	*TT (ms)*	*TPI (ms)*	*Kinematics Score*	*Dynamics Score*
ADMM	12.43	6.329	0.509	1.7543, 0.3727	1.9386, 1.7415
Active-Set	3	3.118	1.039	1.7457, 0.3726	1.9385, 1.7415
IPM	4	0.022	0.005	1.7455, 0.3726	1.9858, 1.7415

* Score : $\hat{x}, \hat{\theta} = \sqrt{\sum (X_r(t) - X(t))^2}$
*CR Convergence Rate * TT Total Time *TPI Time per iteration

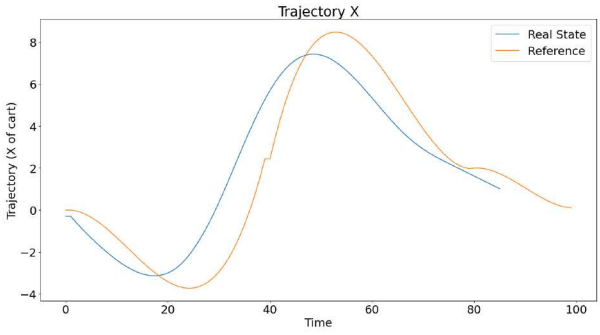

Fig. 2. Fig. illustrates the application of control input on model system. The orange line is the reference trajectory on position of cart and the blue line is the path followed by the cart

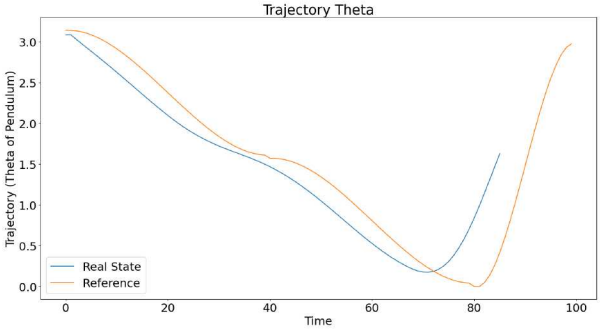

Fig. 3. Fig. also illustrates the application of control input on model system. The orange line is the reference trajectory on the angle of pendulum and the blue line is the angle prescribed for the pendulum to follow. The control input is calculated using an MPC controller

VII. Conclusion

Since we used the qpSWIFT package as the IPM solver, we draw our conclusion by essentially comparing the performance of ADMM and Active-Set solvers. Based on the results obtained on the two applications SVM and MPC, we conclude the following:

1) ADMM has the slowest convergence rate compared to Active-Set and IPM solvers. It has significantly higher convergence rate for SVM and nearly four times the convergence rate for MPC problem compared to Active-Set Method.
2) The total time taken to converge to the optimal solution is highest for ADMM followed by Active-Set Method and IPM. But, ADMM takes less time to complete one iteration compared to the Active-Set solver.
3) The accuracy of each solver is equal to 1.0 in SVM which is because we used a well-separable dataset for evaluation. In the MPC, the score varies slightly for each solver but is less for Active-Set solver compared to ADMM. The least score is obtained for IPM solver. This implies that Active-Set method has better accuracy than ADMM for the applications that we evaluated them on.

References

[1] Gill, Philip & Wong, Elizabeth. (2010). Methods for convex and general quadratic programming. Mathematical Programming Computation. 7. 10.1007/s12532-014-0075-x.

[2] A. G. Pandala, Y. Ding and H. -W. Park, "qpSWIFT: A Real-Time Sparse Quadratic Program Solver for Robotic Applications," in IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3355-3362, Oct. 2019, doi: 10.1109/LRA.2019.2926664.

[3] D. Arnström, A. Bemporad and D. Axehill, "A Dual Active-Set Solver for Embedded Quadratic Programming Using Recursive LDL^T Updates," in IEEE Transactions on Automatic Control, vol. 67, no. 8, pp. 4362-4369, Aug. 2022, doi: 10.1109/TAC.2022.3176430.

[4] Ferreau, Joachim & Kirches, Christian Potschka, Andreas & Bock, Hans Diehl, Moritz. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. Mathematical Programming Computation. 6. 10.1007/s12532-014-0071-1.

[5] B. Stellato, G. Banjac, P. Goulart, A. Bemporad and S. Boyd, "OSQP: An Operator Splitting Solver for Quadratic Programs," 2018 UKACC 12th International Conference on Control (CONTROL), Sheffield, UK, 2018, pp. 339-339, doi: 10.1109/CONTROL.2018.8516834.

[6] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2, 3, Article 27 (April 2011), 27 pages. https://doi.org/10.1145/1961189.1961199

[7] Messikh, Lotfi, El-Hadi Guechi, and Sašo Blažič. 2022. "Stabilization of the Cart–Inverted-Pendulum System Using State-Feedback Pole-Independent MPC Controllers" Sensors 22, no. 1: 243. https://doi.org/10.3390/s22010243

[8] University of Michigan - https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulumsection=SimulinkModeling

[9] L. Vandenberghe http://www.seas.ucla.edu/ vandenbe/236C/lectures/pd.pdf

[10] IRIS Dataset https://archive.ics.uci.edu/ml/datasets/iris