SECOND-ORDER MULTIPLIER UPDATES TO ACCELERATE ADMM METHODS IN OPTIMIZATION UNDER UNCERTAINTY

Jose S. Rodriguez, Gabriel Hackebeil, John Siirola, Victor M. Zavala, Carl D. Laird*
Purdue University – West Lafayette IN
Sandia National Laboratories – Albuquerque NM

Abstract

There is a need for efficient optimization strategies to efficiently solve large-scale, nonlinear optimization problems. Many problem classes, including design under uncertainty are inherently structured and can be accelerated with decomposition approaches. This paper describes a second-order multiplier update for the alternating direction method of multipliers (ADMM) to solve nonlinear stochastic programming problems. We exploit connections between ADMM and the Schur-complement decomposition to derive an accelerated version of ADMM. Specifically, we study the effectiveness of performing a Newton-Raphson algorithm to compute multiplier estimates for the method of multipliers (MM). We interpret ADMM as a decomposable version of MM and propose modifications to the multiplier update of the standard ADMM scheme based on improvements observed in MM. The modifications to the ADMM algorithm seek to accelerate solutions of optimization problems for design under uncertainty and the numerical effectiveness of the approaches is demonstrated on a set of ten stochastic programming problems. Practical strategies for improving computational performance are discussed along with comparisons between the algorithms. We observe that the second-order update achieves convergence in fewer unconstrained minimizations for MM on general nonlinear problems. In the case of ADMM, the second-order update reduces significantly the number of subproblem solves for convex quadratic programs (QPs).

Keywords

Decomposition, Augmented Lagrangian, ADMM, Nonlinear programming.

1. Introduction

Large-scale optimization models are used in many fields of science and engineering to provide solutions to problems. In particular, as uncertainty analysis becomes increasingly important in the areas of optimal design and manufacturing, there is a need for reliable and efficient methods to solve large-scale optimization under uncertainty problems. This has motivated extensive research in the field of decomposition algorithms since they allow considerable computational speedup in parallel computing environments while addressing memory requirements. The alternating direction method of multipliers (Boyd et al., 2011) has received particularl attention due to its flexibility and ease of implementation for solving large-scale problems in a distributed matter. Being a first-order method, one drawback of ADMM is that it has a slow rate of convergence compared to advanced second-order methods. The Schur-complement decomposition has also received special attention for solving stochastic programming problems that arise in design under uncertainty (Biegler, 2017). Schur-complement schemes have proven to be very effective for parallelizing the interior-point algorithm by taking advantage of block-structures in the KKT system of

* Corresponding author: lairdc@purdue.edu

stochastic programming problems (Gondzio et al, 2009, Kang et al., 2014). Although ADMM has a slower convergence rate, it also has reduced communication requirements in parallel computing environments. In this work we derive a second-order multiplier update for ADMM that resembles the Schur-complement algorithm to accelerate its rate of convergence while keeping its ease of implementation and parallel computing benefits. This modification to the ADMM algorithm seeks to make the method more efficient for nonlinear optimization under uncertainty problems.

In section 2 of this paper notation and optimization problem are introduced. Section 3 presents and overview of MM and ADMM for solving general nonlinear problems. It should be mentioned that the methods presented in Section 3 are local methods, in that they do not find global solutions. Section 4 presents a derivation of a second-order multiplier update for MM and ADMM. Connections between the second-order update and the Schur-complement decomposition are highlighted. Section 5 reviews the set of ten stochastic programming problems and provides references. In section 6 results for the ten case studies are provided. Section 7 closes the paper with concluding remarks.

2. Problem Definition

In this paper we consider a problem of the form,

$$\begin{aligned} \min_{x_i,z} \quad & \sum_{i\in P} f_i(x_i) \\ s.t. \quad & c_i(x_i) = 0, && i \in P \\ & A_i x_i + B_i z = 0, (y_i) && i \in P \end{aligned} \tag{1}$$

Where the vector $x_i \in \mathbb{R}^{n_{x_i}}$ contains all the variables corresponding to one particular block i in the set of partitions $P \coloneqq \{1, \dots, N\}$. We define the entire set of block variables $x = (x_1, \dots, x_{|P|})$. The vector $z \in \mathbb{R}^{n_z}$ contains the complicating variables that induce coupling between blocks. The vector of block constraints is represented by $c_i(x_i) \in \mathbb{R}^{m_i}$ and the linking constraints by $A_i x_i + B_i z = 0$ with Lagrange multipliers $y_i \in \mathbb{R}^{n_z}$. We also define the set of block multiplier variables associated with the linking constraints $y = (y_1, \dots, y_{|P|})$.

For convenience we write problem (1) in the following compact form

$$\begin{aligned} \min_{x\in\chi,z} \quad & f(x) \\ s.t. \quad & Ax + Bz = 0, (y) \end{aligned} \tag{2}$$

where $\chi = \{x | c(x) = 0\}$ and $c(x) = (c_1(x_1), \dots, c_{|P|}(x_{|P|}))$. We define the sets $\chi_i = \{x | c_i(x_i) = 0\}$ so that $\chi = \cap_{i\in P} \chi_i$ The matrices A and B are constructed with the block mapping matrices A_i and B_i

This structure is common in large-scale optimization problems. Examples are ubiquitous in engineering problems such as network problems, optimal control problems, parameter estimation problems, and machine learning problems (Word et al., 2014, Zavala et al., 2008). This structure also arises design problems under uncertainty as multi-stage stochastic programming problems. In all cases formulation (1) is valid, and the mapping matrices A_i and B_i determine the type of application of interest. For the particular case of two-stage stochastic problems, like those presented in Section 5, the coupling variables correspond to first-stage variables of the stochastic problem resulting in mapping matrices B_i being the identity matrix.

The separable structure of problem (1) enables the implementation of decomposition techniques that can be used for distributed and parallel computing to avoid memory limitations and/or accelerate the solution of the optimization problem. These decomposition techniques rely on partitioning the problem to exploit the inherent separable structure. Such partitioning can be done externally at the problem level formulation or internally at the linear algebra level of a host algorithm. Partitioning the problem externally is in general less intrusive, but typically exhibits linear or sublinear convergence rates. Partitioning the problem internally can provide improved convergence rates since the properties of the host algorithm are usually retained, however, these approaches are more intrusive and often require significantly more communication in parallel implementation. The goal of this research work is to accelerate the convergence of an external decomposition approach like ADMM by incorporating ideas from an internal decomposition approach like the Schur-complement interior-point algorithm (Kang et al., 2014).

3. Alternating Direction Method of Multipliers

The method of multipliers was first proposed in 1969 by Hestenes (1969). The method seeks to solve NLP (1) by solving a sequence of problems of the form

$$\min_{x,z} \mathcal{L}_\rho(x, z, y^k) = f(x) + {y^k}^T(Ax + Bz) + \frac{1}{2}\rho\left||Ax + Bz|\right|^2 \tag{3}$$

where $\rho > 0$ is a penalty parameter and $\{y^k\}$ is a sequence of multiplier estimates that converges to y. Hestenes proposed generating the sequence of multiplier estimates according to a dual ascent method

$$y^{k+1} = y^k + \alpha\nabla\phi(x^{k+1}, z^{k+1}) \tag{4}$$

where α is a step size, $\phi = \inf\{f(x) + y^T(Ax + Bz)\}$ is the dual function, and $\nabla\phi = Ax^{k+1} + Bz^{k+1}$ its gradient. MM then performs a sequence of minimizations of the augmented Lagrangian function on the primal variables (x, z) and updates the Lagrange multiplier estimates with:

$$(x^{k+1}, z^{k+1}) = \operatorname*{argmin}_{x,z} \mathcal{L}_\rho(x, z, y^k) \tag{5}$$

$$y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1}) \quad (6)$$

Extensive research has been done on the convergence properties of MM (Birgin, 2014). Mangasarian (1975) and Rockafellar (1973) have analyzed the method of multipliers from the perspective of dual theory and demonstrated local convergence for convex and nonconvex problems. Bertsekas (1976) also studied the convergence of MM and showed that, given a sufficiently large penalty parameter $\rho \geq \rho^*$, MM exhibits a linear convergence rate. Moreover, if $\rho \to \infty$ MM exhibits superlinear convergence.

The ease of implementation together with its well understood convergence properties makes MM an attractive algorithm for solving NLP (1). However, the augmented Lagrangian function $\mathcal{L}_\rho(\mathrm{x}, \mathrm{z}, \mathrm{y}^\mathrm{k})$ is not separable, which means that MM cannot be used for decomposition even when solving separable problems like (1). The alternating direction method of multipliers is an inexact version of MM that enables decomposition for separable problems. This is based on the key observation that minimizing over the primal variables x and z separately enables the solution of subproblems over each block i ∈ P. The standard ADMM scheme consist of three updates

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}}\, \mathcal{L}_\rho\left(x_i, z^k, y_i^k\right) \forall\, i \in P \quad (7)$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}}\, \mathcal{L}_\rho(x^{k+1}, z, y^k) \quad (8)$$

$$y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1}) \quad (9)$$

Both methods ADMM and MM have a common underlying algorithmic structure as seen in the updating formulas. However, ADMM blends ideas from MM and Gauss-Seidel schemes to exploit the separable structure of problem (1) via decomposition. Unfortunately, ADMM typically exhibits a sublinear rate of convergence. This has motivated research to design techniques and modifications to accelerate this method. In this work we explore the idea of using a second-order update formula in the space of the multipliers. Interestingly, this second-order multiplier update resembles the multiplier update performed in Schur-complement based algorithms (Kang et al., 2014). Section 4 presents a derivation of the second-order update.

4. Second-order Multiplier Update

In view of the interpretation of the multiplier update as a dual ascent method, it is natural to consider a Newton approach for MM and ADMM. The update formula using a Newton based approach is the following

$$y^{k+1} = y^k - [\nabla^2\phi(x^{k+1}, z^{k+1})]^{-1}\nabla\phi(x^{k+1}, z^{k+1}) \quad (10)$$

where $\nabla^2\phi(x^{k+1}, z^{k+1})$ is the Hessian of the dual function. Miele (1971), Bertsekas (1977, 1982), Fletcher (1975), and others (Betts, 1977, Rupp, 1975, Glad, 1979) studied the local and global convergence properties of applying (10) to the standard MM. Miele (191) was the first to carry numerical experiments using both dual ascent and Newton update formulas for the multipliers on MM. Fletcher (1975) provided numerical evidence for the acceleration obtained with Newton updates together with a comparison with different penalty functions. Rupp (1975) and Bertsekas (1976) prove local convergence for the Newton step using second-order sufficient conditions. Buys and Tapia (1977) later proposed Quasi-Newton update formulas to carry out the multiplier update and overcome the difficulty of obtaining second-order derivative information.

Provided that the subproblems (7) are solved with modern nonlinear solvers (e.g. `Ipopt`, `Knitro`, `SNOPT`) second-order derivative information is readily available. Therefore, in this paper, we explore the application of second-order updates within the ADMM framework.

To derive the second-order update formula consider using Newton method to solve the subproblem

$$\begin{aligned} \min_{x_i} \quad & f_i(x_i) + \rho\left|\left|A_i x_i - B_i z^k\right|\right|^2 & \\ s.t. \quad & c_i(x_i) = 0, & i \in P \\ & A_i x_i + B_i z^k = 0, (y_i) & i \in P \end{aligned} \quad (11)$$

The system of necessary conditions for the *i-th* subproblem are as follows:

$$\nabla f_i(x_i) + A_i^T\left(y + \rho\left(A_i x_i + B_i z^k\right)\right) + \nabla c_i(x_i)^T\lambda_i = 0 \quad (12)$$

$$c_i(x_i) = 0 \quad (13)$$

$$A_i x_i + B_i z^k = 0 \quad (14)$$

Then, given a current iterate $(x_i^k, \lambda_i^k\, y_i^k)$, one obtains the next iterate $(x_i^{k+1}, \lambda_i^{k+1},\ y_i^{k+1})$ with the solution of the linear system of equations:

$$\begin{bmatrix} K_i & E_i^T \\ E_i & \end{bmatrix}\begin{bmatrix} \Delta w_i^k \\ \Delta y_i^k \end{bmatrix} = -\begin{bmatrix} r_i^k \\ A_i x_i^k + B_i z^k \end{bmatrix} \quad (15)$$

where $w_i^k = [x_i^k \quad \lambda_i^k]$ is the vector of the subproblem variables,

$$K_i = \begin{bmatrix} \nabla_{x_i}^2\mathcal{L}\left(x_i^k, \lambda^k, y^k\right) + \rho A_i^T A_i & \nabla_{x_i} c(x_i^k)^T \\ \nabla_{x_i} c(x_i) & \end{bmatrix},$$

is the KKT system of the subproblem, $r_i^k = \left[\nabla_{x_i}\mathcal{L}_\rho^k \quad c(x_i^k)\right]^T$ is the right-hand side for the KKT system of the subproblem, and $E_i = [A_i \quad 0]$ is a compression matrix that extracts the primal coupling variables from the vector of subproblem variables w_i^k. If K_i is invertible one can then solve system (15) explicitly. We first write (15) as

$$K_i\Delta w_i^k + E_i^T\Delta y_i^k = -r_i^k \quad (16)$$

$$E_i\Delta w_i^k = -\left(A_i x_i^k + B_i z^k\right) \quad (17)$$

Solving for Δw_i^k in (16) and substituting in (17) gives the second-order step

$$\Delta y_i^k = [E_i^T K_i^{-1} E_i]^{-1}\left[A_i x_i^k + B_i z^k - E_i^T K_i^{-1} r_i^k\right] \quad (18)$$

Equation (18) provides then a second-order step for updating the multiplier estimates in ADMM.

$$y^{k+1} = y^k + \alpha \Delta y^k \quad (19)$$

Moreover, K_i is directly available in factored form from the solution of the subproblem in (7).

It can be shown that (18) is equivalent to the Schur-complement update of the linking constraint multipliers of a parallel SQP algorithm. The Schur-complement decomposition in Kang et al. (2014) (an internal decomposition) achieves superlinear convergence by applying second-order updates on the y and z variables at a QP level. The second-order update on z is known to add overhead and increase communication in the decomposition algorithm. We explore using only second-order updates on y externally at a problem level with MM and ADMM. We note that Equation (18) is equivalent as well to the Newton update proposed by Bertsekas (1982). Differently than Bertsekas formula, (18) considers a partial elimination of constraints and eliminates only the linear constraints corresponding to the linking across blocks. In his work, Bertsekas (1982) demonstrated three properties of his second-order update formula: 1) the threshold level $\rho \geq \rho^*$ for the second-order update is lower than that for the first-order update; 2) The second-order update has a faster rate of convergence than the first order update; and 3) convergence of the second-order update is guaranteed only for a limited region of initial multipliers. These three properties hold for (18), and hence it can be expected to be less sensitive to the value of ρ, faster to converge, and more dependent on y^0. Section 6 describes how to exploit these properties within an ADMM scheme.

5. Benchmark Problems

We study the performance of the second-order update on MM and ADMM on a set of ten stochastic programming problems. The first six problems are convex QPs of the form:

$$\begin{aligned} \min_{x_i,z} \quad & \sum_{i\in P} p_i (x_i^T Q_i x_i + g_i^T x_i + d_i) \\ s.t. \quad & M_i x_i = b_i, && i \in P \\ & A_i x_i - z = 0, (y_i) && i \in P \end{aligned} \quad (20)$$

Here the data defining the problem is given by the coefficients Q_i, g_i, d_i, M_i; the right-hand side coefficient b_i; and the probabilities p_i. We consider stochastic variants of problems obtained from Biegler (2010) and Kang et al. (2014), and the CUTEr and QPLIB libraries.

Table 1. Benchmark QPs.*

Name	Source	n	n_z	m
SimpleQP	L. Biegler	61	1	60
KangQP	J. Kang	20105	5	16740
CVXQP3	CUTEr	20006	6	15120
AUGD2	CUTEr	4254	14	2200
QPLIB3775*	QPLIB	3600	8	1360
QPLIB0018*	QPLIB	1005	5	120

We generate a stochastic version of the problems by defining Q_i and b_i as a random vector. The normally distributed random Q_i and b_i were generated using the deterministic Q and b as mean. For all problems the scenario probability p_i was set to $1/|P|$. All problems considered a set of 20 scenarios ($|P| = 20$). Statistics for the set of QPs are summarized in Table 1. These QPs provide benchmark problems to compare against the Schur-complement decomposition.

The remaining set of four problems consisted of stochastic non-convex nonlinear DAE constrained optimization problems of the form:

$$\begin{aligned} \min_{s_i,v_i,u_i,z} \quad & \sum_{i\in P} p_i \int_\Gamma J_i(s_i(t), v_i(t), u_i(t), \kappa_i) \\ s.t. \quad & F_i(\dot{s_i}(t), s_i(t), v_i(t), u_i(t), \kappa_i) = 0 && i \in P, \ t \in \Gamma \\ & u_i(t) - z(t) = 0 && i \in P, \ t \in \Gamma \end{aligned} \quad (21)$$

where $\Gamma = [\mathrm{t_0}, \mathrm{t_f}]$ is the time domain; $s_i(t), v_i(t)$ and $u_i(t)$ the state, algebraic and control variables of the *i-th* scenario; κ_i the a vector of time invariant parameters; and F_i the differential algebraic system of equations (DAE).

The differential algebraic system is discretized by applying a collocation on finite element scheme as described by Nicholson, et al. (2017) In the scheme considered the time dependent variables are approximated using Lagrange polynomials defined at a set of collocation points $t_j, j \in \aleph_c := \{0, \dots, n_c - 1\}$ on finite elements $\aleph_e := \{0, \dots, n_e - 1\}$ resulting on a discretized problem with the following NLP form:*

$$\begin{aligned} \min \quad & \sum_{i\in P} p_i \sum_{j\in\aleph_e}\sum_{k\in\aleph_c} q_k \, J_i(s_{i,j,k}, v_{i,j,k}, u_{i,j,k}, \kappa_i) \\ s.t. \quad & F_i(\dot{s_{i,j,k}}, s_{i,j,k}, v_{i,j,k}, u_{i,j,k}, \kappa_i) = 0 && i \in P, j,k \in \aleph_e\aleph_c \\ & A_i x_i - z = 0 && i \in P \end{aligned} \quad (22)$$

* Quadratic problems from the QPLIB database were convexified accordint to $Q + \delta I$

where q_k are the Radau quadrature weights and $x_i = (s_{i,j,k}, v_{i,j,k}, u_{i,j,k})$ is the composition vector containing the subproblem variables. The A_i matrix is constructed to map the control variables $u_{i,j,k}$ in each subproblem to the complicating variables z. For this second set of problems we considered variants of DAE problems available as examples in the `pyomo.dae` package. The problems are labeled according with the name given to the `pyomo.dae` example. As with the set of QPs, we generate the stochastic version of the problems with random vectors κ_i. For additional description of the optimization models see (Nicholson, et al., 2017, Rodriguez et al, 2018)

Table 2. Benchmark DAE constrained problems.

Name	Source	n	n_z	m
OptimalControl	Pyomo	6493	32	6460
RxnKinetics	Pyomo	16161	1	16160
Distillation	Pyomo	62011	30	61980
Semibatch	Pyomo	56524	4	56520

6. Results and Discussion

The optimization problems of Section 5 were implemented using the open-source modeling framework `pyomo`. Pyomo facilitated the discretization of the DAE problems as well as the derivative information necessary for the second-order update. For the discretization of DAE problems we used `pyomo.dae` and for first and second-order derivatives we used the newly introduced package `pyomo.contrib.pynumero`. The solution of the optimization problems was obtained using the state-of the art solver `Ipopt`. For comparison and validation purposes the stochastic NLPs were solved following three different approaches. In the first approach, the original extensive-form was solved directly with `Ipopt`. The results from this approach provided validation for the other approaches. The second approach solved the problems with the method of multipliers. Both, first-order and second-order update formulas were used. In the third approach the problems were decomposed and solved with ADMM. Again, both the first-order and second-order update formulas were used and compared. In all three approaches `Ipopt` was used for solving the corresponding subproblems. We thus highlight that all solutions obtained were local minimizers.

To study the performance of the first and second-order updates the optimization problems were solved with MM and ADMM. The convergence criteria to stop the MM and ADMM iterations are given by

$$\rho\left|\left|Ax^k + Bz^k\right|\right| \leq \epsilon_1 \quad (23)$$

$$\rho A^T B(z^{k+1} - z^k) \leq \epsilon_2 \quad (24)$$

as described in Boyd et al. (2011) and Rodriguez et al. (2018). We used ϵ_1, ϵ_2 =1.0x10^{-6} for all problem instances.

In Table 3 we compare the performance of the method of multipliers with first and second-order updates. To this end, we compare the number of iterations needed to satisfy the convergence criteria in (23) and (24). To overcome the sensitivity of the second-order update on y^0 we perform first-order updates initially if necessary. The switching criteria from first to second-order updates was based on (24). If $\rho A^T B(z^{k+1} - z^k) \leq$ 1.0x10^{-4} then the second-order was used. As we can see for all problem instances MM with a second-order update required significantly fewer iterations. Moreover, for all QPs the second-order update converged in 2-3 iterations resembling the performance of full-space second-order methods on convex quadratic problems.

Table 3. Number of iterations MM.

Name	Penalty Parameter ρ	First order	Second order
SimpleQP	1.0x10	32	2
KangQP	1.0x10^2	67	2
CVXQP3	1.0x10^5	531	2
AUGD2	1.0x10	49	3
QPLIB3775	1.0x10^3	62	3
QPLIB0018	1.0x10^3	126	2
OptimalControl	1.0x10^3	84	6
RxnKinetics	1.0	8	3
Distillation	1.0x10	36	5
Semibatch	1.0x10	18	4

Table 4 summarizes the results for the two multiplier updates on ADMM. For all problems the complicating variables estimates z^0 were initialized following the aggregation initialization strategy of the Progressive Hedging algorithm (Rockafellar et al., 1991). As we can see, the second-order update again reduces significantly the number of iterations for the QPs. However, for the general nonlinear DAE problems the second-order update seems to have little to no effect. The benefits seen in the quadratic problems are attributed to fast convergence of z^k. When ADMM approaches the solution variables z^*, the second-order update accelerates the convergence of the multiplier estimates y^k and the primal infeasibility (23) goes quickly to zero. Differently than MM, ADMM uses a first-order update for the z variables resulting in the better performance observed in Tables 3. This suggests potential benefits of modifying the z-update on ADMM to also follow a second-order approach. This also indicates that the formation of the Schur-complement within an SQP framework may be avoided by solving the KKT system with ADMM and the Newton update proposed here.

Table 4. Number of iterations ADMM.

Name	Penalty Parameter ρ	First order	Second order
SimpleQP	1.0x10	32	2
KangQP	$1.0x10^2$	67	2
CVXQP3	$1.0x10^5$	531	2
AUGD2	1.0x10	49	32
QPLIB3775	$1.0x10^3$	62	28
QPLIB0018	$1.0x10^3$	126	24
OptimalControl	$1.0x10^3$	86	86
RxnKinetics	1.0	66	63
Distillation	1.0x10	49	48
Semibatch	1.0x10	15	13

7. Conclusions

In this work we have presented a second-order multiplier update for ADMM. This update strategy follows a Newton extrapolation to accelerate convergence of multiplier estimates. We have demonstrated the that the Newton update dramatically reduces the number of subproblem solves for convex QPs. These results are evidence that the Newton update of multipliers resembles that of the Schur-complement decomposition. As part of future work we will investigate the performance of ADMM with Newton updates for complicating variables as well as extensions for Newton updates to handle variable bounds.

8. References

D. Bertsekas. Multiplier Methods: A Survey. Automatica, 12(7):133–145, 1976

D. Bertsekas. Approximation procedures based on the method of multipliers. Journal of Optimization Theory and Applications, 23:487–510, December 1977.

D. Bertsekas. Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series). Athena Scientific, 1 edition, 1982.

J. T. Betts. An Accelerated Multiplier Method for Nonlinear Programming. Journal of Optimization Theory and Applications, 21(February):137–174, 1977.

L. T. Biegler. Nonlinear programming concepts, algorithms, and applications to chemical processes. Society for Industrial and Applied Mathematics SIAM, Philadelphia, Pa., 2010.

L. T. Biegler. New nonlinear programming paradigms for the future of process optimization. AIChE J. 63 (4), 1178–1193. 2017

E. G. Birgin and J.M. Martinez. Practical augmented Lagrangian methods for constrained optimization, volume 10. SIAM, 2014.

S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine learning, 3(1):1–122, 2011.

R. Fletcher. An ideal penality function for constrained optimization. J.Inst.Maths Applics, 15(Jan- uary 1974):319–342, 1975.

S. T. Glad. Properties of updating methods for the multipliers in augmented Lagrangians. Journal of Optimization Theory and Applications, 28(2):135–156, 1979.

W. E. Hart, C. D. Laird, J. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. Pyomo–optimization modeling in python, volume 67. Springer Science and Business Media, second edition, 2017.

M. R. Hestenes. Multiplier and gradient. Journal of optimization theory and applications. Vol 4, 5, 1969.

J. Kang, Y. Cao, D. P. Word, and C. D. Laird. An interior-point method for efficient solution of block-structured NLP problems using an implicit Schur-complement decomposition. Computers and Chemical Engineering, 71:563–573, 2014.

J. Gondzio and A. Grothey. Exploiting structure in parallel implementation of interior point methods for optimization. Computational Management Science, 6(2):135–160, May 2009.

O. L. Mangasarian. Unconstrained Lagrangians in nonlinear programming. SIAM Journal on Control, 13(4):772–791, July 1975.

A. Miele. Use of the Augmented Penalty Function in Mathematical Programming Problems) Part 1. Journal of Optimization Theory and Applications, 8(2), 1971.

B. Nicholson, J. D. Siirola, J. Watson, V. M. Zavala, and L. T. Biegler. pyomo.dae: a modeling and automatic discretization framework for optimization with differential and algebraic equations. Mathematical Programming Computation, Dec 2017.

R. T. Rockafellar. The multiplier method of Hestenes and Powell applied to convex programming. Journal of Theory and Applications, 12(6):555–562, November 1973.

R. T. Rockafellar and R. J. Wets. Scenarios and policy aggregation in optimization under uncertainty. Mathematics of operations research, 16(1):119–147, 1991.

J. S. Rodriguez, B. Nicholson, C. D. Laird and V. M. Zavala. Benchmarking ADMM in nonconvex NLPs, Computers and chemical engineering 119:315–324, 2018.

R. Rupp. On the combination of the multiplier method of Hestenes and Powell with Newton's method. Journal of Optimization Theory and Applications, 15(2):169–187, February 1975.

R. A. Tapia. Diagonalized multiplier methods and quasi-Newton methods for constrained optimization. Journal of Optimization Theory and Applications, 22(2):135–194, 1977.

D. P. Word. Efficient parallel solution of large-scale nonlinear dynamic optimization problems. Comput. Optim. Appl. 59 (3), 667–689. 2014

V. M. Zavala, C. D. Laird, L. T. Biegler. Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. Chem. Eng. Sci. 63 (19), 4834–4845. 2008