ALADINp Manual

Alexander Engelmann

October 16, 2019

Contents

1. Introduction

The algorithm is based on the paper [?]. Few algorithmic extensions for numerical stability, e.g. a specific implementation for choosing the Hessian approximation. Furthermore, slightly different problem formulation for numerical reasons. We only consider the local version of ALADIN here, a globally convergent version is topic

2. Problem Formulation

The ALADIN solver solves problems of the form

$$\min_{x} \sum_{i=1}^{N} f_i(x_i) \tag{1a}$$

$$\text{subject to} \quad g_i(x_i) = 0 \qquad \forall i \in \mathcal{R}, \tag{1b}$$

$$h_i(x_i) \leq 0 \qquad \forall i \in \mathcal{R}, \tag{1c}$$

$$\underline{x}_i \leq x_i \leq \bar{x}_i \qquad \forall i \in \mathcal{R}, \tag{1d}$$

$$\sum_{i \in \mathcal{R}} A_i x_i = b, \tag{1e}$$

with $x = (x_1, \ldots, x_N)$ and $\mathcal{R} = \{1, \ldots, N\}$. The objective functions $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$, the constraint functions $g_i : \mathbb{R}^{n_{xi}} \to \mathbb{R}^{n_{gi}}$ and $h_i : \mathbb{R}^{n_{xi}} \to \mathbb{R}^{n_{hi}}$ can possibly be non-convex. In constrast to [?], we distinguish here between (1b), (1c) and (1d) as this leads to more efficient numerical treatment.

3. Solver Interface

The input data for the solver are f_i, g_i, h_i, A_i, b, $\underline{x}_i$, $\bar{x}_i$, $\Sigma_i \succ 0$ and struct called opts.

4. Software Structure

4.1. Solving the Local NLPs

The minimum requirement for local convergence of ALADIN are local solvers and a suitable QP solver.

parameter	data type	typical values
ρ^0	numeric	$10^0 \dots 10^4$
r_ρ	numeric	$1 \dots 2$
μ^0	numeric	$10^3 \dots 10^6$
r_μ	numeric	$1 \dots 2$
`locSol`	string	{ `ipopt`, ... }
`solveQP`	string	{`MA57, linsolve, ...` }
`plot`	logical	-
`Sig`	string	{`Hess, const` }

Table 1: ALADIN options struct.

- `run_ALADIN`
  - `mFun2casFun`
  - `createLocalSolvers`
    - solve local NLPs
    - detect active set
    - evaluate sensitivities
    - `lineSearch`
    - set up coordination QP
    - `solveQP`
    - logging/plotting

Figure 1: ALADIN solver flowchart.

Algorithm 1 Augmented Lagrangian Alternating Direction Inexact Newton (ALADIN)

Initialization: Initial guess (z^0, λ^0), choose $\Sigma_i, \rho^0, \mu^0, \epsilon$.

Repeat:

1. *Parallelizable Step:* Solve for each $i \in \mathcal{R}$

$$\min_{x_i \in [\underline{x}_i, \overline{x}_i]} f_i(x_i) + (\lambda^k)^\top A_i x_i + \frac{\rho^k}{2} \left\| x_i - z_i^k \right\|_{\Sigma_i}^2 \text{ s.t.} \quad h_i(x_i) = 0 \quad \mid \kappa_i^k \tag{2}$$

2. *Termination Criterion:* If $\left\| \sum_{i \in \mathcal{R}} A_i x_i^k \right\| \leq \epsilon$ and $\left\| x^k - z^k \right\| \leq \epsilon$, return $x^\star = x^k$.

3. *Sensitivity Evaluations:* Compute and communicate local gradients $g_i^k = \nabla f_i(x_i^k)$, Hessian approximations $B_i^k \approx \nabla^2 \{ f_i(x_i^k) + \kappa_i^\top h_i(x_i^k) \}$ and constraint Jacobians $C_i^k = \nabla h_i(x_i^k)$.

4. *Consensus Step:* Solve the coordination QP

$$\begin{aligned}
\min_{\Delta x, s} \quad & \sum_{i \in \mathcal{R}} \left\{ \frac{1}{2} \Delta x_i^\top B_i^k \Delta x_i + {g_i^k}^\top \Delta x_i \right\} + (\lambda^k)^\top s + \frac{\mu^k}{2} \|s\|_2^2 \\
\text{s.t.} \quad & \sum_{i \in \mathcal{R}} A_i (x_i^k + \Delta x_i) = s \quad \mid \lambda^{\mathrm{QP}}, \\
& C_i^k \Delta x_i = 0 \qquad\qquad \forall i \in \mathcal{R},
\end{aligned} \tag{3}$$

   obtaining Δx^k and λ^{QP} as the solution.

5. *Line Search:* Update primal and dual variables by

$$z^{k+1} \quad \leftarrow \quad z^k + \alpha_1^k (x^k - z^k) + \alpha_2^k \Delta x^k \qquad\qquad \lambda^{k+1} \leftarrow \lambda^k + \alpha_3^k (\lambda^{\mathrm{QP}} - \lambda^k),$$

   with $\alpha_1^k, \alpha_2^k, \alpha_3^k$ from [**?**]. If full step is accepted, i.e. $\alpha_1^k = \alpha_2^k = \alpha_3^k = 1$, update ρ^k and μ^k by

$$\rho^{k+1} \ (\mu^{k+1}) = \begin{cases} r_\rho \rho^k \ (r_\mu \mu^k) & \text{if } \rho^k < \bar{\rho} \ \ (\mu^k < \bar{\mu}) \\ \rho^k \ (\mu^k) & \text{otherwise} \end{cases}.$$

4.1.1. Active Set Detection

We use a primal active set detection here which considers a constraint $j \in \{1, \dots, n_{hi}\}$ to be active in the current iterate if $h_{i,j}(x_i) > \epsilon_a$ where ϵ_a is a small numerical threshold (e.g. $\epsilon_a = 10^{-6}$). There are other possibilities to do so, e.g. a dual active set detection [?]. The interdependence between the active set strategy, this threshold and also the numerical solver is not fully understood yet and topic of ongoing research. In practice it can happen, that ALADIN fails to identify the correct active set and jumps back and forth between them, cf. subsection 6.1.

4.1.2. Solving Local NLPs Efficiently

In order to avoid numerical difficulties, we consider equality constraints g here explicitly. This avoids unnecessary detentions of active sets for them.

The bound constraints (1d) are principally covered by (1c). However, some solvers [?] can exploit their simplicity to consider them very efficiently and speeding up the local computations.

4.2. Coordination QP

The Lagrangian for (3) is

$$L = \sum_{i \in \mathcal{R}} \left\{ \tfrac{1}{2} \Delta x_i^\top H_i^k \Delta x_i + {g_i^k}^\top \Delta x_i \right\} + {\lambda^k}^\top s + \tfrac{\mu}{2} \|s\|_2^2 + \lambda^{QP\top} (A(x^k + \Delta x) - s - b) + \kappa^{QP\top} C^{act} \Delta x.$$

Thus, the first order optimality conditions are

$$\nabla L = \underbrace{\begin{pmatrix} H & 0 & A^\top & {C^{act}}^\top \\ 0 & \mu I & -I & 0 \\ A & -I & 0 & 0 \\ C^{act} & 0 & 0 & 0 \end{pmatrix}}_{:= M_{KKT}} \begin{pmatrix} \Delta x \\ s \\ \lambda^{QP} \\ \kappa^{QP} \end{pmatrix} - \underbrace{\begin{pmatrix} -g \\ -\lambda^k \\ -Ax^k + b \\ 0 \end{pmatrix}}_{.= m_{KKT}} \overset{!}{=} 0.$$

As $H \succ 0$ and C^{act} and A are assumed to have full rank, this QP has always a unique solution. This QP is then solved in `solveQP` or `solveQPdec` by either direct or iterative methods.

Note that if $H \succ 0$ and C^{act} has full rank, M_{KKT} is invertible. If this is not the case, special care has to be taken to ensure these conditions.

4.3. Regularization

We have some degree of freedom in choosing the Hessian approximations H^k. In any case, we have to make sure that it is positive definite to guarantee convergence of ALADIN and to make sure that the coordination QP has a solution. In practice it may occur that H^k has negative and zero eigenvalues. Thus we use a certain *regularization* procedure to ensure positive definiteness of H^k. The method we propose here is one heuristic which

solver	algorithm	sparse?
`linsolve`		
`MA57`		
`backslash`		
`MOSEK`		
`quadprog`		
`linsol`		

Table 2: List of QP solvers.

worked well in practice for our tested problems. More research is needed here to come up with more systematic procedures.

In order to detect and modify zero and negative eigenvalues, we use an eigenvalue decomposition

$$H^k = \ V\Lambda V^\top$$

where the rows of V are the eigenvectors of H^k and Λ is a diagonal matrix with the corresponding eigenvalues of H^k. By modifying the eigenvalues in Λ yielding to $\tilde{\Lambda}$, we can generate an approximate Hessian $\tilde{H}^k = V\tilde{\Lambda}V^\top \succ 0$. There are different ways of modifying Λ in the literature. One very common approach for the zero eigenvalues is to set them to a small number δ in the range of $10^{-4} \dots 10^{-10}$. For the negative eigenvalues, a similar approach is often used [**?**]. However, this did not work well for many cases and we follow a different approach here: We "flip" the sign of the negative eigenvalues which leads to increasingly smaller stepsizes in the corresponding direction with increasingly negative crvature. This approach worked very well in practice, however, we would like to emphasize that this is just a heuristic and not based on depper theoretical considerations.

4.4. The `solveQP` Subroutine

The `solveQP` subroutine solves equality-constrained QPs of the form

$$\begin{aligned} &\min_x \frac{1}{2} x^\top H x + g^\top x \\ &\text{s.t. } Ax = b. \end{aligned}$$

There is a variety of QP solvers which are interfaced and listed in page 6. For big problems, it is particularly important wether the QP solver is able to exploit sparsity. This is only the case for some of them.

4.5. The `solveQPdec` Subroutine

The `solveQPdec` subroutine solves the coupling QP in a decentralized fashion. In order to do so, it follows the procedure described in [**?**].

4.6. Experimental Features

4.6.1. Line Search

So far, ALADIN usually only comes with local convergence guarantees, at least for non-convex problems. The globalization routines proposed in [?] relies on solving a centralized optimization problem, which is in general not desired (and in some cases even not possible) to compute centrally. As a heuristic, we use the L_1-merit function

$$m(x, \bar{\kappa}) := \sum_{i \in \mathcal{R}} f_i(x_i) + \bar{\kappa}\, \|g_i(x_i)\|_1 + \bar{\kappa}\, \|\max(h_i(x_i), 0)\|_1$$

which is well-known from SQP methods here.[1, 2] With that, we can at least get a sufficient decrease in the coordination step.

The rationale behind merit functions for globalization is quite simple: During the optimization problems, one would like to get "more optimal" and "more feasible" at the same time. These two goals can be expressed in a function by summing up the objective function value plus a factor $\bar{\kappa}$ times some norm of the constraint violation. Thus, by achieving a decrease in m, we can at the same time get a decrease in the objective function value and the constraint violation.

From the definition of m one can see, that all local minimizes of (1) are also local minimizers of m. However, unfortunately not all local minimizers of m are necessarily minimizers of (1).[3] Nonetheless, this merit function is the basis for many globalization routines in context of SQP and we also use it here for determining the step size in the QP step. The underlying SQP theory says, that if we have a positive definite Hessian approximation B_i and certain constraint qualifications are satisfied, then there exists an $\alpha \in (0,\ 1)$ and the update rule $x^+ = z + \alpha \Delta x$ such that we get a sufficient decrease in m [?]. With that it is actually possible to guarantee convergence to stationary points of m for SQP methods. However, as mentioned before, this might in general not be a minimizer of (1), so we only "hope" that this is the case.

In case of ALADIN, even thi guarantee can in general not be given for the full ALADIN step. We can only (similar as in SQP) guarantee, that we get a sufficient decrease in the merit function in the QP step. However, as we still have the decentralized step (and this step does not necessarily produce a decent direction), we can not apply the merit function to the full step and hence also the guarantee for convergence to a stationary point of m does not hold. However, practice has shown that in some cases using a step size rule at least for the coordination can improve convergence. Further research is strongly needed here.

Add example improving convergence here?

[1] Here the inequality constraints h_i also include the bounds (1d) for simplicity.

[2] Note that we neglect the consensus constraint (1e) in m as it is satisfied for any portion of Δx.

[3] See [?] for a counterexample.

4.6.2. Lambda Initialization

4.6.3. $\Sigma_i = H_i$

4.6.4. Nonlinear Slacks

5. Numerical Examples

5.1. Problem Setup with Different Tools

5.1.1. MATLAB Symbolic

```
restoredefaultpath;
clear all;
clc;

addpath('../src');
addpath(genpath('../tools/'))
import casadi.*

%% define Alex's non-convex problem
N    =    2;
n    =    2;
m    =    1;
y1   =    sym('y1',[n,1],'real');
y2   =    sym('y2',[n,1],'real');

f1   =    2*(y1(1)-1)^2;
f2   =    (y2(2)-2)^2;


h1   =    1-y1(1)*y1(2);
% h1   =    [1-y1(1)*y1(2);
%            -1+y1(1)*y1(2)];
h2   =    -1.5+y2(1)*y2(2);

A1   =    [0, 1];
A2   =    [-1,0];
b    =    0;

lb1 =    [0;0];
lb2 =    [0;0];
```

5.1.2. MATLAB Functions

5.1.3. CasADi Symbolic

5.2. Using ALADIN for MPC

5.3. Using ALADIN for OPF

5.3.1. Particular Numerical Issues in OPF

The objective function only depends only on a very small part of the decision vector, namely on p_g. Furthermore,

6. Known Numerical Issues

6.1. Cycling

A. Additional Code

A.1. The `run_ADMM` routine

The Alternating Direction of Multipliers Method (ADMM) seems to be some kind of "state-of-the-art" algorithm for distributed optimization. As it is often used as a benchmark for other algorithms and also shares some conceptual ideas with ALADIN, we included an implementation of this algorithm to the package. We tried to make the interface of the `run_ADMM` routine as similar as possible to the `run_ALADIN` routine, such that the same problem setups can be used.

We use the ADMM version of [?] here.

B. Problem Reformulations

B.1. Consensus Reformulations in Form of (1)

Problems can be reformulated in form of (1) quite easily. Let us consider a "centralized" problem formulation

$$\min_x \sum_{i=1}^{N} f_i(x) \tag{4a}$$

$$\text{subject to} \quad g(x) = 0 \tag{4b}$$

where we only consider equality constraints for simplicity. Now introduce N copies of $x = z_1 = \cdots = z_N$. Then we can write (4) as

$$\begin{aligned} \min_{z_1,\ldots,z_N} \; & \sum_{i=1}^{N} f_i(z_i) \\ \text{subject to} \quad & g_i(z_i) = 0, \qquad i \in \mathcal{R} \\ & z_1 = z_2 = \cdots = z_N \end{aligned}$$

where the constraint functions g_i are an arbitrary partitioning of $g(x) = (g_1(x), \ldots, g_N(x)$ which is in form of (1). This approach is somewhat impactical as it increases the number of decision variables by a factor of N, Therefore, in practice often only a certain subset of the entries of x are copied which couple the individual subsystems commonly leading to a much smaller increase of the problem size.