Morenet documentation

Tillmann Mühlpfordt

May 4, 2020

Abstract

This documents sketches the idea to solve the distributed power flow problem as a distributed nonlinear least-squares problem.

1 Problem formulation

The mathematical formulation for the decentralized power flow problem reads

$$g_i^{\text{pf}}(x_i, z_i) = 0, \tag{1a}$$

$$g_i^{\text{bus}}(x_i) = 0, \tag{1b}$$

$$\sum_{i=1}^{n^{\text{reg}}} A_i \begin{bmatrix} x_i \\ z_i \end{bmatrix} = 0, \tag{1c}$$

where the consensus matrices $A_i \in \mathbb{R}^{4n^{\text{conn}} \times (4n_i^{\text{core}} + 2n_i^{\text{copy}})}$ enforce equality of the voltage angle and the voltage magnitude at the copy buses and their respective original buses. Mathematically speaking, Problem 1 is a system of nonlinear equations; there are as many equations as there are unknowns. In principle, Problem 1 can be solved by Newton's method. However, we would like to explore alternatives to Newton's method, such as distributed optimization.

2 Problem solution

Currently, we solve Problem 1 as a distributed *feasibility* probem, namely

$$\min_{x_i, z_i \; \forall i \in \{1,\dots,n^{\text{reg}}\}} 0 \quad \text{s. t.} \tag{2a}$$

$$g_i^{\text{pf}}(x_i, z_i) = 0, \tag{2b}$$

$$g_i^{\text{bus}}(x_i) = 0, \tag{2c}$$

$$\sum_{i=1}^{n^{\text{reg}}} A_i \begin{bmatrix} x_i \\ z_i \end{bmatrix} = 0. \tag{2d}$$

From our experience so far, we can say that Aladin is a viable method, but ADMM is not. Solving a distributed feasibility problem, however, is not the only way. We can also think of solving Problem 1 as a

Table 1: Our experience so far with solving Problem 1. A "—" indicates further investigations are required.

Method	Solver	Number of iterations	Wall clock time	Remark
ADMM	Casadi & Ipopt	many	not acceptable	
Aladin	Casadi & Ipopt	few	acceptable	does not scale well to larger problems ($N \approx 100$)
	fmincon & Jacobian	few	acceptable	tends to become slow for larger problems ($N \approx 300$)
	Ipopt & Jacobian	—	—	—
Least squares	—	—	—	—

distributed least-squares problem of the form

$$\min_{x_i, z_i \; \forall i \in \{1,\dots,n^{\text{reg}}\}} \left\| \begin{bmatrix} g_i^{\text{pf}}(x_i, z_i) \\ g_i^{\text{bus}}(x_i) \end{bmatrix} \right\|^2 \quad \text{s. t.} \quad \sum_{i=1}^{n^{\text{reg}}} A_i \begin{bmatrix} x_i \\ z_i \end{bmatrix} = 0. \tag{3a}$$

Clearly, the solution to Problem 2 is the solution to Problem 3 (why?). Hence, we can think of solving the least-squares Problem 3 as a necessary condition.

2.1 Next steps

The goal is to explore how to solve the distributed power flow problem via a distributed least-sqaures Problem 3. To do so, there are a couple of immediate next steps

- Familiarize with nonlinear least-squares (Gauss-Newton, Levenberg-Marquardt, etc)
- Solve a prototypical distributed power flow problem.
- Use the Jacobian matrix that is given analytically as a return value from `generate_distributed_problem`.
- Work out how the Aladin algorithm simplifies in the absence of both equality and inequality constraints. (Note that inequality constraints in the form of lower/upper bounds might still be necessary, i.e. $\underline{x} \leq x \leq \overline{x}$)
- Exploit the problem structure as much as possible. As a general rule, the bigger the problem, the more it pays off to exploit sparsity etc.

Table 1 lists the experience we have gained thus far. The goal is to fill out everything that is related to least squares.

Another thing to explore is to interface Ipopt directly from Matlab. This is somewhat painful to set up but might be worth the speed up anyhow.

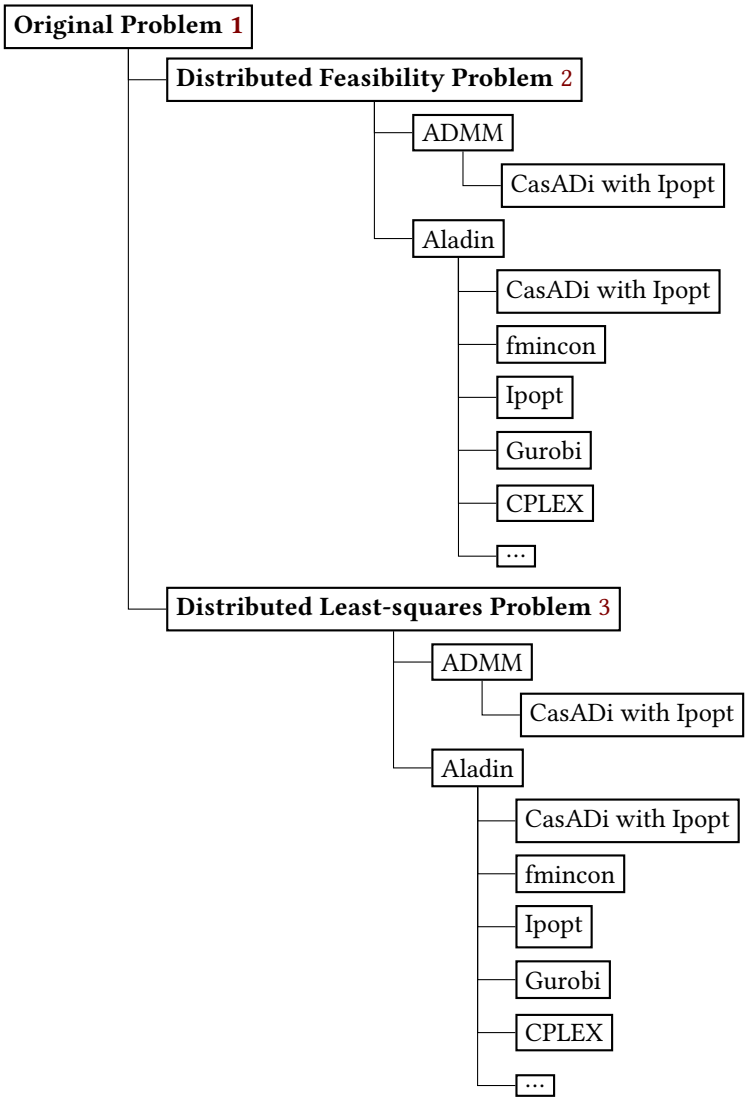

Figure 1: Algorithm tree