POPDIP:
a POsitive-variables Primal-Dual Interior Point method

Ed Bueler

December 6, 2018

Abstract

The algorithm documented here is a version of a Newton-type primal-dual interior point algorithm in [2], namely Algorithm 16.1 in section 16.7. The version here minimizes a smooth nonlinear function subject to the simple constraints that all the variables are nonnegative. That is, it is a specialized algorithm for nonnegativity constraints; it is not suitable for general inequality constraints.

These short notes are not research! This algorithm is simply a special case of a well-known algorithm. Furthermore "POPDIP" is just a name I made up; it is not in common use! However, this class of algorithms is new to me so I am documenting it fully.

Introduction

Consider a nonlinear optimization problem with nonnegativity (informally: positivity) constraints on the variables:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \geq 0 \end{array} \tag{1}$$

As usual, "$x \geq 0$" means that each entry of $x \in \mathbb{R}^n$ is nonnegative. The feasible set for (1) is the convex and closed set $S = \{x \in \mathbb{R}^n : x \geq 0\}$ with interior $S^\circ = \{x \in \mathbb{R}^n : x > 0\}$.

Here $f : \mathbb{R}^n \to \mathbb{R}^n$ is a continuous and smooth function. In fact this algorithm only assumes f is defined and smooth on S°, but in practice, so as to have a chance of good performance, f should be well-behaved near the boundary of S.

One can start the derivation by considering a logarithmic barrier function. Let $\mu > 0$. If $x \in S^\circ$ then the following function is well-defined and finite:

$$\beta_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \ln x_i \tag{2}$$

Let $\{e_1, \dots, e_n\}$ be the standard basis of $\mathbb{R}^n$. The first-order necessary condition for the unconstrained problem of minimizing β_μ, namely $\nabla \beta_\mu(x) = 0$ for $x \in S^\circ$, is

$$\nabla f(x) - \mu \sum_{i=1}^{n} \frac{1}{x_i} e_i = 0 \tag{3}$$

Conditions (3) can be reformulated by defining additional variables

$$\lambda_i = \frac{\mu}{x_i}$$

so that $\lambda \in \mathbb{R}^n$. Note that $\lambda > 0$ if and only if $x > 0$ because $\lambda_i x_i = \mu > 0$. Then (3), plus feasibility for x, is precisely equivalent to the following nonlinear system of equations and inequalities:

$$\begin{aligned} x &\geq 0 \\ \lambda &\geq 0 \\ \nabla f(x) - \lambda &= 0 \\ \lambda_i x_i &= \mu, \qquad i = 1, \ldots, n \end{aligned} \tag{4}$$

The feasible set for x and for λ is the same, namely $S \subset \mathbb{R}^n$. Because of the last condition in (4), both x and λ are positive and thus in the interior S°. By contrast, for the general primal-dual interior point Algorithm 16.1 [2, section 16.7], the feasible set for the primal variable x is different from the dual feasible set for λ. For example, generally the dimension is different.

The third condition in (4) can be written using a Lagrangian function for (1), namely

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i=1}^{n} \lambda_i x_i, \tag{5}$$

in which case it states that $\nabla_x \mathcal{L}(x, \lambda) = 0$. However, system (4) describes a solution which is different from the KKT conditions for (1). (See Lemma 14.8 and Theorem 14.18 in [2].) Those KKT conditions are the same as system (4) but with the last equation replaced by

$$\lambda_i x_i = 0, \qquad i = 1, \ldots, n \tag{6}$$

That is, complementary slackness applies in the KKT conditions. System (4) modifies the KKT conditions by connecting the variables through equality with a positive constant ($\lambda_i x_i = \mu$).

Also, the stationary-point conditions for Lagrangian (5), namely $\nabla_x \mathcal{L}(x, \lambda) = 0$ and $\nabla_\lambda \mathcal{L}(x, \lambda) = 0$, do not generally apply because we have nonnegativity constraints on x and *not* equality constraints (which would make the problem trivial). Of course $x_i = 0$ holds when the ith constraint is active, but the algorithm here will not keep track of the active set.

Algorithm design

Algorithm 16.1 [2] applies to (1), and the POPDIP algorithm proposed below is the simplification which uses the fact that $g_i(x) = x_i$. All such primal-dual interior point algorithms compute approximate solutions to systems like (4) for a sequence $\{\mu_k\}$ going to zero. In the limit where $\mu = 0$, which is never achieved because the algorithm is terminated after a finite number of steps, the solution will solve (6).

Each step of the algorithm is a Newton step for the equalities in (4), namely

$$\begin{aligned} \nabla f(x) - \lambda &= 0 \\ \lambda_i x_i &= \mu_k, \qquad i = 1, \ldots, n. \end{aligned} \tag{7}$$

The Newton method updates both x and λ using the linearization of these equations. Because of the second equation, these equations are always nonlinear. Note that if f is quadratic then the first equation is linear.

To describe the Newton step let $x = x_k + \Delta x$ and $\lambda = \lambda_k + \Delta\lambda$. We assume that the current iterate (x_k, λ_k) does not solve (7). The unknowns in the Newton step for the search direction $p = (\Delta x, \Delta\lambda)$. Substituting into (7) and expanding to first order gives

$$\begin{aligned} \nabla f(x_k) + \nabla^2 f(x_k)\Delta x - \lambda_k - \Delta\lambda = 0 \\ (\lambda_k)_i (x_k)_i + (x_k)_i (\Delta\lambda)_i + (\lambda_k)_i (\Delta x)_i = \mu_k, \qquad i = 1, \ldots, n \end{aligned} \tag{8}$$

Rearranging as a linear block system for the search direction, and suppressing the current-iterate subscript, we get the Newton step equations

$$\begin{bmatrix} \nabla^2 f(x) & -I \\ \Lambda & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + \lambda \\ -\Lambda x + \mu_k e \end{bmatrix} \tag{9}$$

Here I is the $n \times n$ identity matrix and the other notation is as follows:

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}, \qquad X = \begin{bmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{bmatrix}, \qquad e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Given a solution of (9) the update formulas are

$$\begin{aligned} x_{k+1} &= x_k + \alpha_P \Delta x \\ \lambda_{k+1} &= \lambda_k + \alpha_D \Delta\lambda \end{aligned}$$

The maximum step sizes α_P, α_D for the primal and dual variables are determined by separate ratio tests to achieve (strict) positivity of x_{k+1} and λ_{k+1}, respectively. Because this is a Newton search one uses $\alpha_P = \alpha_D = 1$ for the largest allowed step. Note we are not truly using $p = (\Delta x, \Delta\lambda) \in \mathbb{R}^{2n}$ as a search direction because of the separate searches for the primal and dual variables. In fact back-tracking is not needed to maintain feasibility of the primal variables because of the linearity of the constraint functions in (1), namely $g_i(x) = x_i$. As on page 645 for the slack variables, a ratio test on x suffices to keep x feasible.

The optimality test is the same as in Algorithm 16.1 [2]. It uses the merit function

$$\nu(x, \lambda) = \max\{\|\nabla f(x) - \lambda\|, \|\Lambda x\|\} \tag{10}$$

where $\|\cdot\|$ denotes the usual L^2 norm on $\mathbb{R}^n$. Note that once $\mu_k \to 0$ we have $\nu(x_*, \lambda_*) = 0$ for the exact optimum, but when $\mu_k \neq 0$ then the exact solution of (7) does not make $\nu(x, \lambda)$ zero. In fact $\nu(x_*, \lambda_*) = \sqrt{n}\,\mu_k$ for the exact solution of (7); if the merit function value $\nu(x_k, \lambda_k)$ is close to $\sqrt{n}\,\mu_k$ then we should decrease μ_k more rapidly.

Subsection 16.7.2 of [2] describes how Algorithm 16.1 should be modified so that, by Theorem 16.17, the algorithm will exhibit the local quadratic convergence one would want from a Newton-type algorithm. Note that quadratic convergence is much harder to achieve in a constrained problem. This is why I am interested in this algorithm.

First of all the general inequality constraints should be augmented by slack (excess) variables which have simple nonnegativity constraints. But in our case the constraints are already of that type! Indeed the general replacement "$g_i(x) \geq 0$" by "$g_i(x) - s_i = 0$, $s_i \geq 0$" is, in our case, the replacement "$x_i \geq 0$" by "$x_i - s_i = 0$, $s_i \geq 0$". This would be a meaningles renaming of the existing variables, so we need not introduce "s_i" in our case.

Secondly, Theorem 16.17 uses a specific barrier parameter update for μ_k and a specific parameter choice for κ in the ratio test. In our case these are equations

$$\begin{aligned} \mu_k &= \min\{\theta\nu(x_k, \lambda_k), \nu(x_k, \lambda_k)^2\}, \\ \kappa &= \max\{\bar{\kappa}, 1 - \nu(x_k, \lambda_k)\}, \end{aligned}$$

for the merit function $\nu(x, \lambda)$ in (10) and for parameters $0 < \theta < 1$ and $0 < \bar{\kappa} < 1$. We choose these parameters based on the minimal hints in Example 16.16 on pages 643–644.

Algorithm

We can now present a pseudocode for our algorithm.

ALGORITHM POPDIP.

inputs primal initial values x_0 such that $x_0 > 0$

smooth function f returning $f(x)$, $\nabla f(x)$, and $\nabla^2 f(x)$

parameters $0 < \texttt{tol}$ [default $\texttt{tol} = 10^{-4}$]

$0 < \mu_0$ [default $\mu_0 = 1$]

$0 < \theta < 1$ [default $\theta = 0.1$]

$0 < \bar{\kappa} < 1$ [default $\bar{\kappa} = 0.9$]

output an estimate (x_k, λ_k) of the solution

- determine initial dual variables: $(\lambda_0)_i = \mu_0/(x_0)_i$
- for $k = 0, 1, 2, \ldots$
  - (i) optimality test: if $\nu(x_k, \lambda_k) < \texttt{tol}$ then stop
  - (ii) barrier parameter: $\mu_k = \min\{\theta\nu(x_k, \lambda_k), \nu(x_k, \lambda_k)^2\}$
  - (iii) compute Newton step by solving this system for $(\Delta x, \Delta\lambda)$:

$$\begin{bmatrix} \nabla^2 f(x_k) & -I \\ \Lambda_k & X_k \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) + \lambda_k \\ -\Lambda_k x_k + \mu_k e \end{bmatrix}$$

  - (iv) ratio test for step sizes to keep x_{k+1}, λ_{k+1} positive:

$$\begin{aligned} \kappa &= \max\{\bar{\kappa}, 1 - \nu(x_k, \lambda_k)\} \\ \alpha_P &= \min_{1 \leq i \leq n} \left\{ 1, -\kappa \frac{(x_k)_i}{(\Delta x)_i} \ : \ (\Delta x)_i < 0 \right\} \\ \alpha_D &= \min_{1 \leq i \leq n} \left\{ 1, -\kappa \frac{(\lambda_k)_i}{(\Delta \lambda)_i} \ : \ (\Delta \lambda)_i < 0 \right\} \end{aligned}$$

(v) the update:

$$x_{k+1} = x_k + \alpha_P \Delta x$$
$$\lambda_{k+1} = \lambda_k + \alpha_D \Delta \lambda$$

This algorithm is implemented by a MATLAB code with signature

```
function [xk,lamk,...]  = popdip(x0,f,tol,mu0,theta,kappabar)
```

Only inputs `x0,f` are required. The parameters have the default values listed above. The suppressed outputs here are `xklist` and `lamklist`, namely lists of the iterates; if they are not requested then they are not saved. Download the code at

bueler.github.io/M661F18/matlab/popdip.m

Testing

We only try an easy 2D test problem in code `testpopdip.m`:

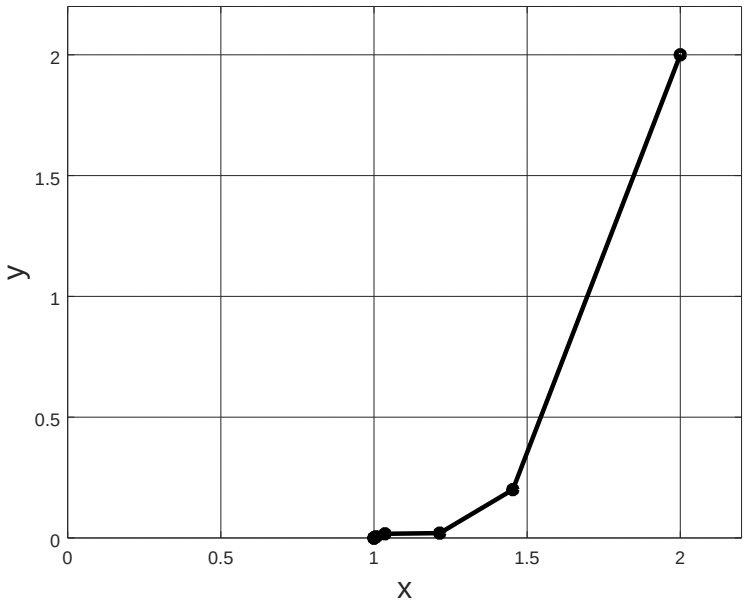
$$\begin{array}{ll} \text{minimize} & f(x) = \frac{1}{2}(x_1 - 1)^2 + \frac{1}{2}(x_2 + 1)^2 \\ \text{subject to} & x \geq 0 \end{array} \tag{11}$$

For this objective function f the unconstrained minimum is the infeasible point $\hat{x} = (1, -1)^\top$. A sketch shows the exact solution of the constrained problem is $x_* = (1, 0)^\top$. We propose to start with the feasible point $x_0 = (2, 2)^\top$. Note the initial dual variables are then determined using $\mu_0 = 1$—thus $\lambda_0 = (1/2, 1/2)^\top$—but from then on the relationship $x_i \lambda_i = \mu_k$ only holds approximately as x_k and λ_k are each updated separately.

Running the code with the given initial iterate x_0 and `tol` $= 10^{-14}$, and otherwise using the default parameters in POPDIP, gives

```
>> testpopdip([2 2]',1.0e-14)
  1:    2.000000000000000    2.000000000000000
  2:    1.453253013076204    0.200000000000000
  3:    1.214566731865968    0.020000000000000
  4:    1.035839709499516    0.017240583591047
  5:    1.006566228657711    0.004849254151787
  6:    1.000109109789261    0.000090069040616
  7:    1.000000031919418    0.000000028128803
  8:    1.000000000000003    0.000000000000003
```

The printed columns are $(x_k)_1$ and $(x_k)_2$. Note the apparent quadratic, or at least strongly superlinear, convergence. The iterates x_k are graphed in the following figure.



Possible improvements

We may consider possible improvements of our algorithm. First, in Algorithm 16.1 the computation of the Newton search direction is followed by separate line searches in x and in λ. These line searches only maintain nonnegativity and they do not seek sufficient decrease of $f(x)$; they only use ratio tests. Secondly, equation (9) can be symmetrized by multiplying the second half of the equations by $-\Lambda^{-1}$:

$$\begin{bmatrix} \nabla^2 f(x) & -I \\ -I & -\Lambda^{-1}X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) + \lambda \\ x - \mu_k \Lambda^{-1} e \end{bmatrix} \tag{12}$$

These facts suggests two possible changes:

1. Back-tracking line search is appropriate as a globalization even for unconstrained optimization. Thus there must be cases where it is appropriate for problem (1) as well. Once the ratio tests are applied, further back-tracking could be used based on sufficient decrease. Compare the modified back-tracking line searches in [1].

2. One can replace linear system (9) with symmetrized system (12).

Determining if these are actual improvements would require testing which we have not done.

References

[1] S. BENSON AND T. MUNSON, *Flexible complementarity solvers for large-scale applications*, Optimization Methods and Software, 21 (2006), pp. 155–168.

[2] I. GRIVA, S. G. NASH, AND A. SOFER, *Linear and Nonlinear Optimization*, SIAM Press, 2nd ed., 2009.