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

Ed Bueler

November 23, 2022

Abstract

The algorithm here is an example of the Newton-type primal-dual interior point algorithms presented in [7, section 16.7], [10, chapter 19], and [13]. It minimizes a smooth nonlinear function subject to the constraints that all the variables are nonnegative, plus linear equality constraints:

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

This is a specialized algorithm for nonnegativity constraints $x \geq 0$, and it is not suitable for general inequality constraints of the form "$g_i(x) \geq 0$." Nor is it suitable for general equality constraints "$g_i(x) = 0$." However, it can be used as an interior-point method for linear programming in the case where $f(x) = c^\top x$, thus (1) is in standard form, though it has no special performance improvements for that case.

These short notes are *not* research. I simply implement a case of a well-known algorithm. Furthermore "POPDIP" is a name I made up; it is not in common use. I am interested in the class of interior point methods because local quadratic convergence can be proven.

Introduction

First consider a nonlinear optimization problem with only nonnegativity (informally: positivity) constraints on the variables:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \geq 0 \end{array} \tag{2}$$

As usual, $x \geq 0$ in (2) means that each entry of $x \in \mathbb{R}^n$ is nonnegative. The feasible set for (2) is the convex and closed set $S = \{x \in \mathbb{R}^n : x \geq 0\}$, with interior $S^\circ = \{x \in \mathbb{R}^n : x > 0\}$, and $f : S \to \mathbb{R}$ is assumed to be a continuous and smooth function.

The next section will address the general problem (1) including linear equality constraints.

One can derive an interior point method for (2) by considering a logarithmic barrier function [7, section 16.2]. Let $\mu > 0$ be a (temporarily) fixed constant. If $x \in S^\circ$ then the following function is well-defined:

$$\beta_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \ln x_i \tag{3}$$

Recalling that $\lim_{x\to 0^+} \ln x = -\infty$, this barrier imposes "infinite cost" to reaching the boundary of the feasible set S.

Now let $\{e_1, \ldots, e_n\}$ 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

$$\nabla f(x) - \mu \sum_{i=1}^{n} \frac{1}{x_i} e_i = 0 \tag{4}$$

Conditions (4) 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 (4), plus feasibility for x, is equivalent to the following nonlinear system of equations and inequalities:

$$\begin{aligned} \nabla f(x) - \lambda &= 0 \\ \lambda_i x_i &= \mu, \qquad i = 1, \ldots, n \\ x &\geq 0 \\ \lambda &\geq 0 \end{aligned} \tag{5}$$

The feasible set for x and for λ is the same, namely $S \subset \mathbb{R}^n$. (By contrast, for the general problem (1), the feasible set for the primal variables x is different from that of the dual variables λ.) Because of the second condition in (5), based on constant $\mu > 0$, both x and λ are strictly positive and thus in the interior S°.

The first condition in (5) can be rewritten using a Lagrangian function for (2):

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i=1}^{n} \lambda_i x_i. \tag{6}$$

Now (5) states that $\nabla_x \mathcal{L}(x, \lambda) = 0$. In fact, the first-order KKT conditions [7, sections 14.4, 14.5] of (2) are nearly the same as (5) but with the second equation replaced by complementarity,

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

Thus the barrier method which generated system (5) modifies KKT conditions (7) by changing the complementary slackness condition into a nonzero connection between the primal and dual variables, with their product is set to a positive constant. System (5) describes a primal and dual (x, λ) solution which is interior to the feasible set, and different from the solution to (7).

The first condition in (7) allows λ to be eliminated if desired: $\lambda = \nabla f(x)$. Writing the KKT conditions without these dual variables gives a *nonlinear complementarity problem* (NCP) [5],

$$x \geq 0, \qquad \nabla f(x) \geq 0, \qquad x_i (\nabla f(x))_i = 0. \tag{8}$$

Thus we may regard POPDIP as solving the μ-modified NCP

$$x \geq 0, \qquad \nabla f(x) \geq 0, \qquad x_i(\nabla f(x))_i = \mu. \tag{9}$$

However, the method achieves quadratic convergence substantially because it updates primal and dual variables separately. Stating the algorithm or the original KKT conditions as an NCP, thereby suppressing the dual variables, is not beneficial to solver performance. On the other hand, such NCP formulations also appear in problems which are not optimizations, rather variational inequalities, including in a glaciological case upon which I am focussed [2].

General problem

Suppose from now on that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously-differentiable. Let $A \in \mathbb{R}^{m \times n}$, $m \leq n$, be a full rank matrix, and suppose $b \in \mathbb{R}^m$. The POPDIP algorithm solves general problems of form (1) given in the abstract and restated here:

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

Observe that the constraints are in standard form for linear programming (LP) problems [7, chapter 4]. If $f(x) = c^\top x$ for some fixed $c \in \mathbb{R}^n$ then (1) is an LP problem in standard form, and then POPDIP becomes an LP interior point method, though one with no special adaptations to that case; compare e.g. [14].

The (primal) feasible set is

$$S = \{Ax = b \text{ and } x \geq 0\} \subset \mathbb{R}^n. \tag{10}$$

The iterates x_k from our algorithm will be in the interior,[1] defined to be

$$S^o = \{Ax = b \text{ and } x > 0\} \subset \mathbb{R}^n. \tag{11}$$

We will assume that S^o is nonempty and that a feasible iterate in S^o can be found. (In fact the user will *not* need to provide an initial iterate which is feasible with respect to "$Ax = b$", but they will need to provide a strictly-positive initial iterate.)

Consider the following Lagrangian for (1):

$$\mathcal{L}(x, \tau, \lambda) = f(x) - \tau^\top (Ax - b) - \lambda^\top x, \tag{12}$$

for multipliers $\tau \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^n$. Function (12) reduces to (6) if there are $m = 0$ linear equality constraints (so τ is absent).

The first-order necessary conditions, the KKT conditions, for (1) are given by $\nabla_x \mathcal{L} = 0$ and $\nabla_\tau \mathcal{L} = 0$, plus complementarity for the primal variables x and dual variables λ:

$$\begin{aligned} \nabla f(x) - A^\top \tau - \lambda &= 0 \\ -Ax + b &= 0 \\ \lambda_i x_i &= 0, \qquad i = 1, \dots, n \\ x &\geq 0 \\ \lambda &\geq 0 \end{aligned} \tag{13}$$

[1] In general, S^o is not the topological interior of $S \subset \mathbb{R}^n$, which is empty when equality constraints apply ($m > 0$).

Conditions (13) reduce to system (7) given in the Introduction when there are $m = 0$ linear equality constraints. Note that the dual feasible set $T = \{\lambda \geq 0\} \subset \mathbb{R}^n$ is generally different from, and larger than, the primal feasible set S.

Algorithm design

The POPDIP algorithm solves (1) by applying a Newton method to a μ-modified form of KKT conditions (13). The modification replaces complementarity $\lambda_i x_i = 0$ by the conditions

$$\lambda_i x_i = \mu_k, \qquad i = 1, \ldots, n$$

for a (scalar) barrier sequence $\{\mu_k > 0\}$ which will go to zero. Thus POPDIP computes primal and dual iterates in the interiors: $x \in S^o$, $\lambda \in T^o$. However, if the method converges as $\mu_k \to 0$ then the limiting values of x and λ solve the KKT conditions (13). Though the algorithm is terminated after a finite number of steps, in examples the KKT system is solved to high accuracy.

Each step of the algorithm is a Newton step for the modified equalities from (13), namely

$$\begin{aligned} \nabla f(x) - A^\top \tau - \lambda &= 0 \\ -Ax + b &= 0 \\ \lambda_i x_i = \mu_k, \qquad & i = 1, \ldots, n \end{aligned} \tag{14}$$

Because of its third equation, (14) is always a nonlinear system. However, if f is quadratic or linear then *only* the third equation, the modified complementarity condition, is nonlinear.

Intuitively, an interior-point iteration for (14) sees "changing equations" with iteration k. As we update the μ_k value, each new iterate is attempting to solve new equations. Thus, though the iterates of POPDIP should converge to the solution of KKT problem (13), they will not converge to a solution of (14) for any particular $\mu_k > 0$.

The Newton step computes a search direction in the x, τ, λ variables using the linearization of equations (14). The actual POPDIP step is subject to positivity constraints on x and λ, via a ratio test [7, section 3.1], uses generally different step sizes for the three parts x, τ, λ.

To describe the Newton step in detail, let $x = x_k + \Delta x$, $\tau = \tau_k + \Delta\tau$ and $\lambda = \lambda_k + \Delta\lambda$. The unknowns form a candidate search direction $p = (\Delta x, \Delta\tau, \Delta\lambda)$ which is modified later. Substituting into (14) and expanding to first order gives

$$\begin{aligned} \nabla f(x_k) + \nabla^2 f(x_k)\Delta x - A^\top \tau_k - A^\top \Delta\tau - \lambda_k - \Delta\lambda &= 0 \\ -Ax_k - A\Delta x + b &= 0 \\ (\lambda_k)_i (x_k)_i + (x_k)_i (\Delta\lambda)_i + (\lambda_k)_i (\Delta x)_i = \mu_k, \qquad & i = 1, \ldots, n \end{aligned} \tag{15}$$

Rearranging this as a vectorized and matrix-block system gives

$$\begin{bmatrix} \nabla^2 f(x_k) & -A^\top & -I \\ -A & 0 & 0 \\ \Lambda_k & 0 & X_k \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\tau \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) + A^\top \tau_k + \lambda_k \\ Ax_k - b \\ \mu_k e - \Lambda_k x_k \end{bmatrix}. \tag{16}$$

Here I is the $n \times n$ identity matrix, $e = (1, 1, \ldots, 1)^\top \in \mathbb{R}^n$, and we have defined $n \times n$ diagonal matrices from the primal/dual vectors $x, \lambda \in \mathbb{R}^n$ as follows:

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

Note that $\Lambda x = X\lambda$ in this notation.[2]

Linear system (16) can also be written as

$$M_k p = -r_k \tag{17}$$

where M_k is the $(2n+m) \times (2n+m)$ square and sparse *system matrix* on the left side of (16) and $r_k = r_k(x_k, \tau_k, \lambda_k) \in \mathbb{R}^{2n+m}$ is the residual of system (14) for the current iterate. That is, for a given $\mu_k > 0$ we may consider the *residual function*

$$r_k(x, \tau, \lambda) = \begin{bmatrix} \nabla f(x) - A^\top \tau - \lambda \\ -Ax + b \\ \Lambda x - \mu_k e \end{bmatrix}. \tag{18}$$

In fact, system (16), or (17), can be made smaller by noting that the diagonal matrix X_k is invertible, because $x_k > 0$ strictly in (14). Thus the last row of system (16) can be solved for $\Delta\lambda$ in terms of Δx:

$$\Lambda_k \Delta x + X_k \Delta\lambda = \mu_k e - \Lambda_k x_k \quad \Longleftrightarrow \quad \Delta\lambda = \mu_k X_k^{-1} e - \lambda_k - X_k^{-1}\Lambda_k \Delta x \tag{19}$$

Instead of (16) we will actually solve the smaller, *symmetric* system

$$\begin{bmatrix} \nabla^2 f(x_k) + X_k^{-1}\Lambda_k & -A^\top \\ -A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\tau \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) + A^\top \tau_k + \mu_k X_k^{-1} e \\ Ax_k - b \end{bmatrix} \tag{20}$$

for $\Delta x, \Delta\tau$, and then complete the Newton step computation by assigning

$$\Delta\lambda = \mu_k X_k^{-1} e - \lambda_k - X_k^{-1}\Lambda_k \Delta x. \tag{21}$$

The Appendix gives a proof that the matrix in (20) is invertible.

We will solve system (20) using direct linear algebra, via Matlab's default "\" linear solver. Because this direct solver is quite accurate, the residual norm $\|b - Ax_k\|$, part of the right-hand side of (20), will be very small. In this sense, system (20) imposes the classical null-space requirement on the (primal) search direction for linear equality constraints [7, chapter 3]:

$$A\Delta x = 0. \tag{22}$$

Note there is no equivalent linear restriction on $\Delta\lambda$; if $\lambda_k \in T^o$ then $\Delta\lambda$ can be anything.

On the other hand, use of a direct solver means the implementation is not scalable to large problem sizes. A scalable implementation would not assemble the full matrix in (20), and it

[2] The product $\Lambda x = X\lambda$ is simply `x.*lambda` in MATLAB.

might use an iterative solver, along the way exploiting the block structure. Preconditioners [3, 12] for the upper-left block, and then for the equality-constraint Schur complement [3, chapters 14] as well, should be considered.

Given a solution $(\Delta x, \Delta\tau, \Delta\lambda)$ of (20) and (21), the update formulas are

$$\begin{aligned} x_{k+1} &= x_k + \alpha_x \Delta x \\ \tau_{k+1} &= \tau_k + \alpha_\tau \Delta\tau \\ \lambda_{k+1} &= \lambda_k + \alpha_\lambda \Delta\lambda \end{aligned} \tag{23}$$

Separate step sizes $\alpha_x, \alpha_\tau, \alpha_\lambda$ are used for the three vector variables x, τ, λ, though this is not the only choice [13]. Because the constraint equations are linear, the full Newton step $\alpha_\tau = 1$ is always used for the τ multipliers.

The quadratic convergence of a general primal-dual interior point method is proven in subsection 16.7.2 of [7]. That method replaces general inequality constraints $g_i(x) \geq 0$ by $g_i(x) - s_i = 0$, with slack variables $s_i \geq 0$. However, in our case this replacement says $x_i - s_i = 0$ and $s_i \geq 0$, clearly unneeded as it simply renames existing primal variables x.

Also, the back-tracking globalization of the inequality constraints described in algorithm 16.1 of [7], choosing α_x from $\{1, 1/2, 1/4, \dots\}$ so that $g_i(x_k + \alpha_x \Delta x) > 0$, is not needed to maintain primal feasibility because of the linearity of the constraint functions in (1). That is, for constraints $Ax = b$ and $x \geq 0$ a ratio test for Δx will guarantee primal feasibility, as also holds in the simplex method [7, chapter 5].

POPDIP will shorten the step in the Newton search direction *only* because of the inequality constraints, and not for the "usual" (unconstrained) line search reasons, for instance to generate sufficient decrease of f by an Armijo or Wolfe rule [7, section 11.5]. In this sense our method is most suitable for quadratic functions f, and it may need additional refinements for more general, especially non-convex, objective functions. It should be used with caution if the Hessian $\nabla^2 f(x)$ varies significantly during the iteration.

POPDIP determines step sizes α_x, α_λ for the inequality-constrained primal and dual variables via a condition of (strict) positivity for x_{k+1} and λ_{k+1}. Suppose $0 < \kappa < 1$. For the primal variables, assuming $x_k \in S^o$ so that $x_k > 0$, we require that for all i such that $(\Delta x)_i < 0$,

$$(x_k)_i + \alpha_x (\Delta x)_i \geq (1-\kappa)(x_k)_i \qquad \Longleftrightarrow \qquad \alpha_x \leq -\kappa \frac{(x_k)_i}{(\Delta x)_i}. \tag{24}$$

(If $(\Delta x)_i \geq 0$ then $(x_k)_i + \alpha_x (\Delta x)_i \geq (1-\kappa)(x_k)_i$ automatically.) Rule (24) is applied with $\kappa \approx 1$, so that the inequality constaints are nearly activated, and also with a preference for the Newton step $\alpha_x = 1$. A similar ratio test applies to the dual variables and determines α_λ. In summary:

$$\begin{aligned} \alpha_x &= \min_{1 \leq i \leq n} \left\{ 1, \; -\kappa \frac{(x_k)_i}{(\Delta x)_i} \; : \; (\Delta x)_i < 0 \right\}, \\ \alpha_\tau &= 1, \\ \alpha_\lambda &= \min_{1 \leq i \leq n} \left\{ 1, \; -\kappa \frac{(\lambda_k)_i}{(\Delta \lambda)_i} \; : \; (\Delta \lambda)_i < 0 \right\}. \end{aligned} \tag{25}$$

(Compare subsection 16.7.2 in [7].)

The optimality test in POPDIP requires that a certain merit function associated to system (14) be smaller than a given tolerance, either in an absolute sense or relative to the initial value of the merit function. We use a merit function which is a norm of the residual (18):

$$\nu(x, \tau, \lambda) = \max\{\|\nabla f(x) - A^\top \tau - \lambda\|_2, \|b - Ax\|_2, \|\Lambda x\|_2\}. \tag{26}$$

This function is fundamentally the same as in section 16.7 of [7], but adapted to our problem. (Recall that we expect $\|b - Ax_k\| = 0$ to good accuracy.) Note that in the limit $\mu_k \to 0$ we have $\nu(x_*, \tau_*, \lambda_*) = 0$ for the exact solution of (14). On the other hand, for particular $\mu_k > 0$ the exact solution of (14) yields $\nu(x_*, \tau_*, \lambda_*) = \sqrt{n}\,\mu_k > 0$.

Theorem 16.17 in [7] describes how algorithm 16.1 should be modified so that the algorithm will exhibit the local quadratic convergence. The Theorem requires a specific scheme for μ_k, and a specific choice for κ in ratio tests (25). In our case these are the equations

$$\mu_k = \min\{\theta\nu(x_k, \tau_k, \lambda_k), \nu(x_k, \tau_k, \lambda_k)^2\}, \tag{27}$$

$$\kappa = \max\{\bar\kappa, 1 - \nu(x_k, \tau_k, \lambda_k)\}, \tag{28}$$

for parameters $0 < \theta < 1$ and $0 < \bar\kappa < 1$. We choose the default values of the parameters as $\theta = 0.1$ and $\bar\kappa = 0.9$, based on the minimal hints in example 16.16 in [7].

The method by which the initial dual variables are determined (below) is a somewhat ad hoc construction for which I have no reference, but compare [6].

Pseudocode

We now present a pseudocode for POPDIP. The defaults for parameters are `rtol` $= 10^{-4}$, `atol` $= 10^{-50}$, `maxiters` $= 200$, $\theta = 0.1$, and $\bar\kappa = 0.9$.

ALGORITHM POPDIP.

inputs smooth function $f : \mathbb{R}^n \to \mathbb{R}$; user code returns $f(x)$, $\nabla f(x)$, and $\nabla^2 f(x)$
full (row) rank matrix A of size $m \times n$ with $0 \le m \le n$
vector $b \in \mathbb{R}^m$
primal initial values (vector) $x_0 \in \mathbb{R}^n$ such that $x_0 > 0$

parameters `rtol` > 0
`atol` > 0
`maxiters` > 0
$0 < \theta < 1$
$0 < \bar\kappa < 1$

output an estimate (x_k, τ_k, λ_k) of the solution to (1) and (13)

- determine initial dual variables:
  (i) $\tau_0 = 0$
  (ii) $g_0 = \nabla f(x_0)$

(iii) if $g_0 \leq 0$ then $\mu_0 = 1$; otherwise μ_0 is average of those products $(x_0)_i(g_0)i$ where $(g_0)_i > 0$

(iv) $(\lambda_0)_i = \mu_0/(x_0)_i$ for $i = 1, \ldots, n$

- for $k = 0, 1, 2, \ldots,$ `maxiters` $- 1$

(i) evaluate gradient $g_k = \nabla f(x_k)$

(ii) evaluate merit function:

$$\nu_k = \max\{\|g_k - A^\top \tau_k - \lambda_k\|_2, \|b - Ax_k\|_2, \|\Lambda_k x_k\|_2\}$$

(iii) optimality test: if $\nu_k <$ `atol` or $\nu_k <$ (`rtol`) ν_0 then stop

(iv) evaluate barrier parameter: $\mu_k = \min\{\theta\nu_k, \nu_k^2\}$

(v) solve linear system (20) for Δx and $\Delta\tau$:

$$\begin{bmatrix} \nabla^2 f(x_k) + X_k^{-1}\Lambda_k & -A^\top \\ -A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\tau \end{bmatrix} = \begin{bmatrix} -g_k + A^\top \tau_k + \mu_k X_k^{-1} e \\ Ax_k - b \end{bmatrix}$$

(vi) evaluate $\Delta\Lambda$:

$$\Delta\lambda = \mu_k X_k^{-1} e - \lambda_k - X_k^{-1}\Lambda_k \Delta x$$

(vii) evaluate ratio test parameter: $\kappa = \max\{\bar{\kappa}, 1 - \nu_k\}$

(viii) apply ratio tests to find step sizes which keep x_{k+1}, λ_{k+1} positive:

$$\alpha_x = \min_{1 \leq i \leq n} \left\{ 1, \; -\kappa \frac{(x_k)_i}{(\Delta x)_i} \; : \; (\Delta x)_i < 0 \right\}$$

$$\alpha_\tau = 1$$

$$\alpha_\lambda = \min_{1 \leq i \leq n} \left\{ 1, \; -\kappa \frac{(\lambda_k)_i}{(\Delta \lambda)_i} \; : \; (\Delta \lambda)_i < 0 \right\}$$

(ix) update to the next iterate:

$$x_{k+1} = x_k + \alpha_x \Delta x, \qquad \tau_{k+1} = x_k + \alpha_\tau \Delta x, \qquad \lambda_{k+1} = \lambda_k + \alpha_\lambda \Delta\lambda$$

MATLAB implementation

The POPDIP algorithm is implemented by a MATLAB code `popdip.m`. If one wants to accept the default parameter values then the signature is

```
function [x,tau,lam] = popdip(x0,f,A,b)
```

where `x` $\in \mathbb{R}^n$, `tau` $\in \mathbb{R}^m$, `lam` $\in \mathbb{R}^n$ are approximations of the solution to (13). The user-provided function `f` has signature

```
function [fx,dfx,Hfx] = f(x)
```

where `fx` $= f(x) \in \mathbb{R}$ is the objective value, `dfx` $= \nabla f(x) \in \mathbb{R}^n$ is the gradient, and `Hfx` $= \nabla^2 f(x) \in \mathbb{R}^{n \times n}$ is the Hessian.

For more control the POPDIP parameters can be set to non-default values as follows:

```
popdip(x0,f,A,b,rtol,atol,maxiters,theta,kappabar)
```

The parameters have the default values listed in the last section. Additional diagnostic outputs can be requested by using more output arguments:

```
[x,tau,lam,iteratelist,nuklist,muklist] = popdip(...)
```

If `iteratelist` is not requested (`nargout<4`) then the `lists` are not stored.

Download the `popdip.m` code, this documentation, and the example codes demonstrated in the next section, from the Github repository: `github.com/bueler/popdip`.

Examples

Example 1.

We first try a small 2D test problem in `small.m`, a positivity-constrained quadratic minimization:

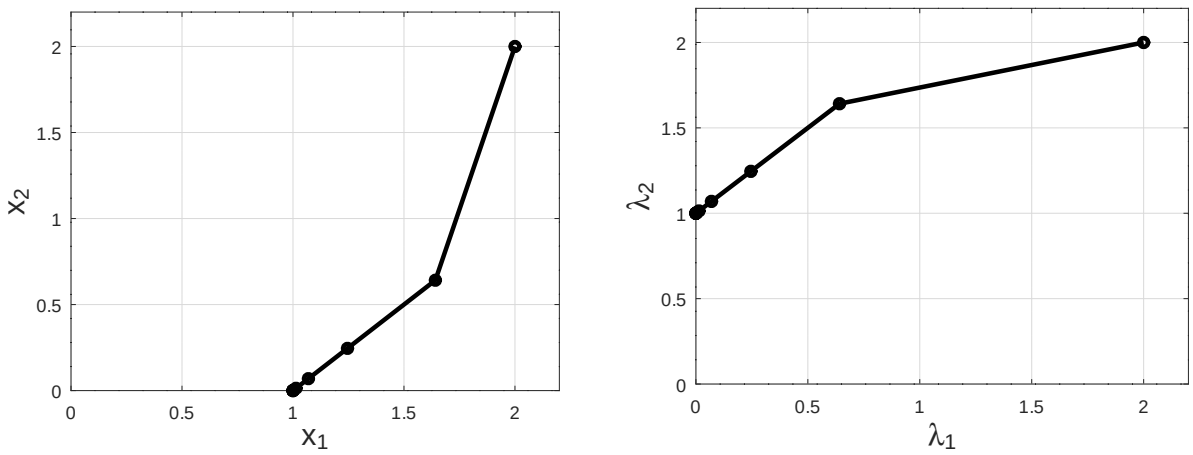
$$\begin{array}{ll} \text{minimize} & f(x) = \frac{1}{2}(x_1 - 1)^2 + \frac{1}{2}(x_2 + 1)^2 \\ \text{subject to} & x \geq 0 \end{array} \tag{29}$$

The unconstrained minimum is the infeasible point $\hat{x} = (1, -1)^\top$. A contour sketch shows that the exact solution of the constrained problem is $x_* = (1, 0)^\top$.

Running with the given initial iterate $x_0 = (2, 2)^\top$ and `rtol` $= 10^{-14}$, and otherwise using the default parameters in POPDIP, gives strongly superlinear, and possibly quadratic, convergence:

```
>> small
        x_1                 x_2                 nu_k                mu_k
  0:    2.000000000000000   2.000000000000000   5.656854249492381
  1:    1.641421356237309   0.641421356237310   1.488944443027284   0.565685424949238
  2:    1.245446520462486   0.245446520462486   0.432311672017441   0.148894444302728
  3:    1.069404818969903   0.069404818969903   0.104965543876699   0.043231167201744
  4:    1.013447008853577   0.013447008853577   0.019272663285475   0.010496554387670
  5:    1.000537794151783   0.000537794151783   0.000760964805668   0.000371435550115
  6:    1.000000867357066   0.000000867357066   0.000001226629190   0.000000579067435
  7:    1.000000000002257   0.000000000002257   0.000000000003192   0.000000000001505
  8:    1.000000000000000   0.000000000000000   0.000000000000000   0.000000000000000
```

Both the merit function values ν_k and the barrier parameters μ_k go rapidly to zero. The primal iterates x_k are graphed in the left figure below, and the dual iterates λ_k at right. We see complementarity at the solution: $x_1\lambda_1 = 0$ since $\lambda_1 = 0$ and $x_2\lambda_2 = 0$ since $x_2 = 0$.



Example 2.

The next problem is an $n = 5$ linear programming (LP) problem, in standard form, with $m = 3$ linear equality constraints:

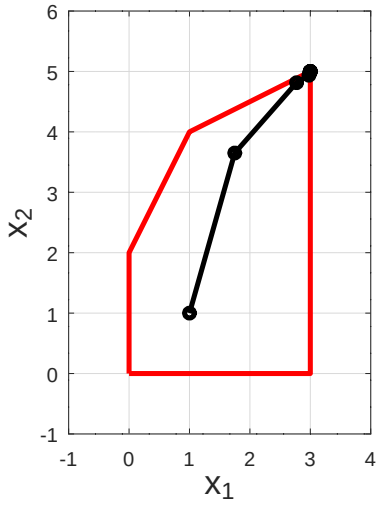
$$\begin{array}{ll} \text{minimize} & f(x) = c^\top x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array} \tag{30}$$

where $c = \begin{bmatrix} -1 & -2 & 0 & 0 & 0 \end{bmatrix}^\top$, $A = \begin{bmatrix} -2 & 1 & 1 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$, and $b = \begin{bmatrix} 2 & 7 & 3 \end{bmatrix}^\top$.

This problem is solved by the simplex method in [7, section 5.2]. Starting from $x = 0$ the simplex method finds the solution on the third iteration.

Standard-form problem (30) arises from 2D LP problem

$$\begin{array}{ll} \text{minimize} & z = -x_1 - 2x_2 \\ \text{subject to} & -2x_1 + x_2 \leq 2 \\ & -x_1 + 2x_2 \leq 7 \\ & x_1 \leq 3 \\ & x_1, x_2 \geq 0 \end{array} \tag{31}$$



This feasible set of (31) can be easily sketched (red outline).

Problem (30) is defined in a short driver program `linear.m`. It runs POPDIP with initial iterate $x_0 = (1, 1, 1, 1, 1)^\top$ and `rtol` $= 10^{-14}$, otherwise using the default parameters. The result is strongly superlinear convergence similar to Example 1:

```
>> linear
        x_1                 x_2                 nu_k                mu_k
  0:    1.000000000000000   1.000000000000000   5.477225575051662
  1:    1.749124060578910   3.650175187884218   2.662463982283214   0.547722557505166
  2:    2.771966360502739   4.813544496010846   0.673064121890067   0.266246398228321
  3:    2.977196636050274   4.937189711618397   0.151356274478520   0.067306412189007
  4:    2.993139910878286   4.988863175677374   0.033061726195566   0.015135627447852
  5:    2.999412769621014   4.999167116475287   0.002445966295331   0.001093077739031
  6:    2.999996717485349   4.999995406776724   0.000013394027090   0.000005982751118
  7:    2.999999999901378   4.999999999862158   0.000000000401658   0.000000000179400
  8:    3.000000000000000   5.000000000000000   0.000000000000000   0.000000000000000
```

The 2D iterates x_k are in black in the figure, apparently good performance of an interior-point method on an LP problem. Compare figure 5.1, chapter 10, and section 10.6 in [7].

It is worth considering the Newton step equations (20) in this LP case. Because the objective function $f(x) = c^\top x$ is linear, the gradient $\nabla f = c$ is constant and the Hessian is zero, so (20) becomes

$$\begin{bmatrix} X_k^{-1}\Lambda_k & -A^\top \\ -A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \tau \end{bmatrix} = \begin{bmatrix} -c + A^\top \tau_k + \mu_k X_k^{-1} e \\ Ax_k - b \end{bmatrix}. \tag{32}$$

The square system matrix on the left of (32) is nontheless invertible when $\mu_k > 0$. (This is proven in the Appendix. One only needs the facts that A has full row rank and that $\mu_k \neq 0$.) Thus it is okay that the `linear.m` code supplies a zero Hessian.

Regarding application of POPDIP to LP problems, one might ask how it does on the famously-bad Klee-Minty cube example [7, chapter 9]. The answer for now is "not any better than simplex"! Presumably the parameters and details could be adjusted to improve the performance.[3] In any case we can compare the results of the `kleeminty.m` example code here to the result of

`bueler.github.io/opt/assets/codes/kleeminty.m`

which uses the simplex method. Both seem to generate exponentially-increasing numbers of iterations as the dimension of the cube increases.

Example 3.

Next we consider a continuum (infinite-dimensional) optimization problem with positivity constraints. The discretized problem has arbitrary size n, but otherwise the problem is like Example 1; it is a quadratic optimization with only positivity constraints.

The continuum problem is an *obstacle problem* [3, chapter 12],

$$\begin{array}{ll} \text{minimize} & f(u) \\ \text{subject to} & u \geq 0 \end{array} \tag{33}$$

where

$$f(u) = \int_0^1 \frac{1}{2} u'(x)^2 - q(x)u(x)\,dx. \tag{34}$$

The admissible functions $u(x)$ in (33) must be continuous and have one well-defined (i.e. L^2) derivative on the interval $0 \leq x \leq 1$. Furthermore we assume boundary conditions $u(0) = u(1) = 0$.

If $y = u(x)$ gives the height of a taught string which is pinned at the ends then we may interpret $\frac{1}{2}u'(x)^2$ in (34) as its elastic energy. This quantity becomes larger as there are more bends in the graph of $u(x)$. In this context the continuous source function $q(x)$ is an external upward/downward force on the string. The reason (33) is called an "obstacle" problem [8] is that the zero function, the obstacle in this case, stops the solution from going further down even if negative values of $q(x)$ would force it down: $u(x) \geq 0$.

Source functions $q(x)$ which are sometimes negative and sometimes positive are of greatest interest because they generate a free boundary at a point $x = \alpha$ where $u(\alpha) = 0$ and $u'(\alpha) = 0$, although $u(x)$ is positive on one side of α. (See the derivation of the exact solution (39) below for a particular case.) However, when $q(x) \leq 0$ everywhere then one may show that $u(x) \equiv 0$ is the global minimizer. On the other hand, if $q(x) \geq 0$ everywhere and $q(x) > 0$ at any one point then $u(x) > 0$ on the whole open interval $(0, 1)$, so the inequality constraint is nowhere active.

The discretization replaces $u(x)$ by a continuous and piecewise-linear function. These functions are linear between the points of a pre-determined grid on $[0, 1]$, and our discretization scheme is a basic finite element approach [4].[4] Let x_i be points $x_0 = 0 < x_1 < x_2 < \cdots < x_n <$

[3] See the discussion of following the central path in chapter 14 of [10].

[4] It can also be interpreted as finite differences [9].

$x_{n+1} = 1$, but equally-spaced for simplicity. Then $\Delta x = 1/(n+1)$ is the spacing and we have $x_i = i\Delta x$. Only the interior grid points are locations of (primal) variables in the optimization problem (33): $u = (u_1, \dots, u_n)^\top \in \mathbb{R}^n$ where $u_i = u(x_i)$.

Now we approximate the objective function (34) using the midpoint numerical integration rule. In the interval (x_i, x_{i+1}) the slope of $u(x)$ is $(u_{i+1} - u_i)/\Delta x$, and the midpoint value of $u(x)$ is $u(x_{i+1/2}) = (u_i + u_{i+1})/2$. Also we assume $q(x)$ is given by a formula which we may evaluate at will. The discrete objective function is, therefore,

$$f_n(u) = \frac{\Delta x}{2} \sum_{i=0}^{n} \left(\left(\frac{u_{i+1} - u_i}{\Delta x} \right)^2 - q(x_{i+1/2})(u_i + u_{i+1}) \right) \tag{35}$$

where the endpoint values are $u_0 = 0$ and $u_{n+1} = 0$ when they appear. (These are notational conveniences and not actual variables.) Observe that (35) is actually a whole class of example objective functions. For each dimension $n = 1, 2, 3, \dots$ there is a finite-dimensional optimization problem derived from the continuum problem (33).

It follows that the gradient is

$$(\nabla f_n(u))_i = \frac{1}{\Delta x} \left\{ \begin{array}{cc} 2u_1 - u_2 & (i = 1) \\ -u_{i-1} + 2u_i - u_{i+1} & (1 < i < n) \\ -u_{n-1} + 2u_n & (i = n) \end{array} \right\} - \frac{\Delta x}{2}(q(x_{i-1/2}) + q(x_{i+1/2})) \tag{36}$$

for $i = 1, \dots, n$. Since the objective function is quadratic, its Hessian is a constant matrix, of a form well-known to be positive-definite [9]:

$$\nabla^2 f_n(u) = \frac{1}{\Delta x} \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & & \ddots & \\ & & -1 & 2 \end{bmatrix} \tag{37}$$

To test the accuracy of the implementation we must find an exact solution $u_{\text{exact}}(x)$ to the continuum problem. This solution will not be an exact solution to any of the discrete problems, so for no n will we know the exact minimizer $u_* \in \mathbb{R}^n$ of $f_n(u)$. However, because our discretization is convergent [4, 9], for large n the finite-dimensional minimizer u_* should be close to the grid values $u_{\text{exact}}(x_i)$. If we solve the $f_n(u)$ optimization problem accurately then our numerical minimizer $u \in \mathbb{R}^n$ should be close to $u_* \in \mathbb{R}^n$, and thus also close to $u_{\text{exact}}(x)$. In any case, we will measure the total numerical error, the norm of the grid point errors $|u_i - u_{\text{exact}}(x_i)|$.

The continuum exact solution $u_{\text{exact}}(x)$ solves (33) for $q(x) = -100(\cos(2\pi x)+0.7)$; this source function is shown in the left plot below. The details of this $q(x)$ are not special except that $q(x)$ is positive on a small interval around $x = 0.5$, specifically on $(0.3734, 0.6266)$. The exact solution $u_{\text{exact}}(x)$ is found by observing that the (strong form) NCP form of problem (33) is

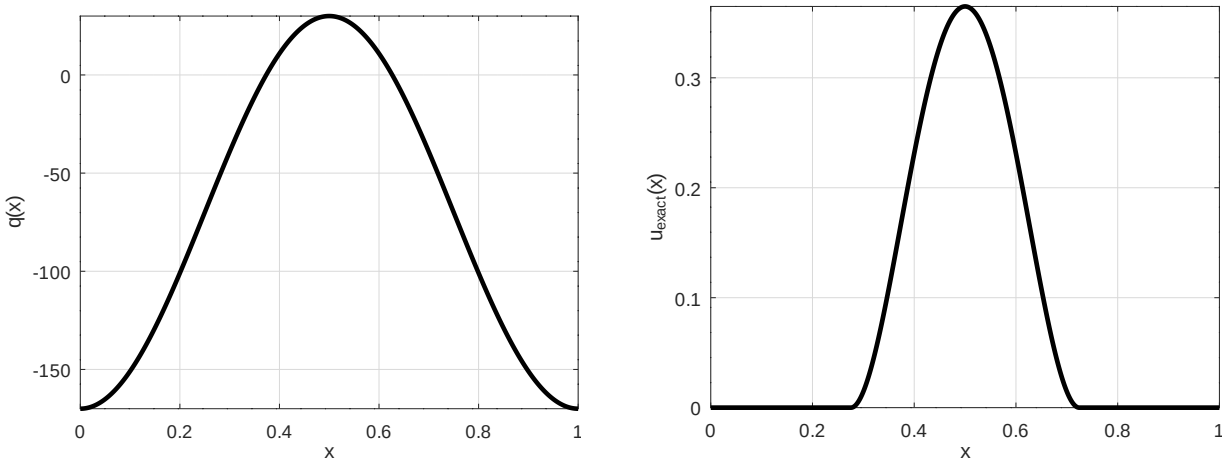
$$u(x) \geq 0, \qquad -u''(x) - q(x) \geq 0, \qquad u(x)(-u''(x) - q(x)) = 0. \tag{38}$$

The solution to (33) and (38) is symmetric around $x = 0.5$ because $q(x)$ is symmetric around that point. A free boundary [8] will arise at a location $0 < \alpha < 0.5$ where $u(\alpha) = 0$ and $u'(\alpha) = 0$.

Combining these facts gives the formula[5]

$$u_{\text{exact}}(x) = \begin{cases} 0, & 0 \le x \le \alpha \\ 100\left(\frac{\cos(2\pi\alpha)-\cos(2\pi x)}{(2\pi)^2} - \frac{0.7}{2}(\alpha(\alpha-1)-x(x-1))\right), & \alpha < x < 1-\alpha \\ 0, & 1-\alpha \le x \le 1. \end{cases} \tag{39}$$

Here $\alpha = 0.275562026630539$ gives the exact (i.e. very accurate) location $x = \alpha$ of the free boundary on the interval $0 < x < 0.5$. By symmetry there is a matching free boundary on $0.5 < x < 1$, namely at $1 - \alpha$.



The exact solution is plotted in the right figure. Note that the set on which $u_{\text{exact}}(x) = 0$, i.e. where the constraint is active, is a strict subset of the set on which $q(x) < 0$, where there is a downward force on the string. Finding the free boundary $x = \alpha$, where the string first meets the obstacle, is a typical goal for this kind of problem.

Formula (39) appears in the short driver program `obstacle.m`. Because the exact solution is known, the numerical error can be measured at the end. The program calls POPDIP with initial iterate $(u_0)_i = 1$, and parameters `rtol` $= 10^{-12}$ and `theta` $= 0.001$, otherwise using the defaults. The user-supplied objective function $f_n(u)$, namely (34), and the gradient, Hessian, and the formula for $q(x)$, all appear in the function `obstaclefcn.m`, the handle of which is also passed to POPDIP.

By default `obstacle.m` uses $n = 25$ interior grid points. The output merit function ν_k values, and the dynamically-adjusted barrier parameters μ_k, suggest strongly superlinear convergence similar to previous Examples:

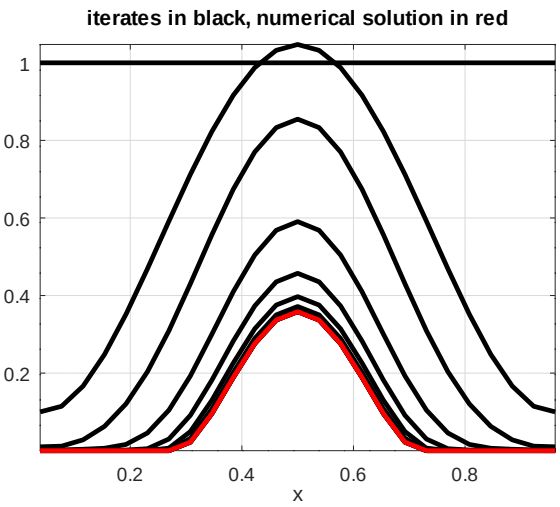
```
>> obstacle
        nu_k                    mu_k
  0:   42.668104033897947
  1:    5.652366439731654     0.042668104033898
  2:    1.582225195585537     0.005652366439732
  3:    0.405205172171047     0.001582225195586
```

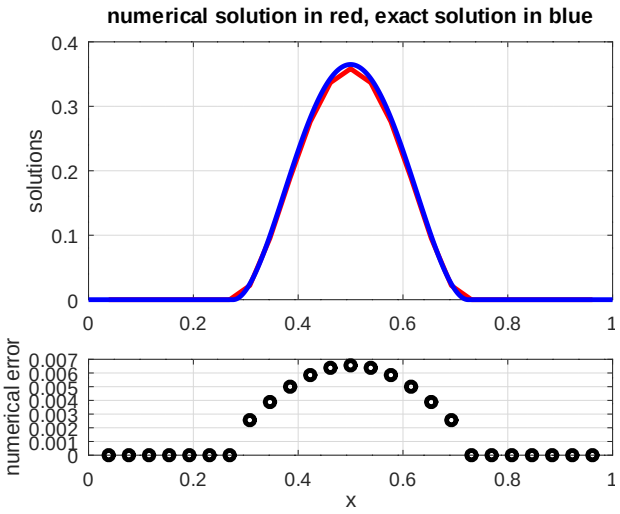
[5]This is a nonobvious calculation. In summary, one starts from $u'' = -q$, and integrates once, setting the constant using symmetry $u'(0.5) = 0$. Then use $u'(\alpha) = 0$ to determine α by solving a transcendental equation: $\sin(2\pi\alpha)/(2\pi) + 0.7(\alpha - 0.5) = 0$. (I used Newton's method to compute α to 15 digits.) Then integrate again to find $u(x)$, and use $u(\alpha) = 0$ to set the constant. The result is (39).

```
  4:     0.244137215303266    0.000405205172171
  5:     0.201522593604293    0.000244137215303
  6:     0.107490917285153    0.000201522593604
  7:     0.029115141695205    0.000107490917285
  8:     0.000205231703144    0.000029115141695
  9:     0.000017959316470    0.000000042120052
 10:     0.000000001669991    0.000000000322537
 11:     0.000000000000004    0.000000000000000
12 iterations with n=25: ||u-uexact||_inf = 6.557e-03
```

The final numerical error norm $\|u - u_{\text{exact}}\|_\infty = 6.557 \times 10^{-3}$ shows that $u \in \mathbb{R}^{25}$ is close to the continuum exact solution at the grid points. On the other hand, our tolerance $\texttt{rtol} = 10^{-12}$ has made u *much* closer to the constrained optimum u_* of $f_{25}(u)$. That is, most of the numerical error is discretization error, which we can only reduce by choosing larger n.[6]

The run also produces the figures below. Note that the initial iterate (left figure) is far from the correct solution, and indeed the merit function value ν_k stagnates for a while before entering the region of superlinear convergence. In the right figure note that the numerical error jumps up from zero abruptly at the free boundary. In fact $u_{\text{exact}}(x)$ has a discontinuous second derivative at the free boundary $x = \alpha$, and thus there is no way for a piecewise-linear function to do a good job of approximating it.





The reader should experiment with larger n values. The numerical error can be reduced by an order of magnitude by two grid-doubling refinements to $n = 100$. However, beyond that the POPDIP algorithm is not so well-behaved, and significantly smaller numerical error is hard to achieve. The reasons are not all clear to me even though I have been working with problems like this for a while [2].

Possible improvements

POPDIP is an implemented primal-dual interior point method, but it is far from the best algorithm which experts could create. The following non-exhaustive list of possible improvements suggests what an expert might focus on.

[6] Or by choosing a different discretization scheme. One might want to use a spectral method, but it will not work easily for obstacle problems!

- The step sizes α can be chosen more carefully so that the iterates x_k, τ_k, λ_k remain in a neighborhood of the central path. Chapter 14 of [10] may be a good reference, but see also [13] and [14]. Section 14.2 of [10] describes a predictor-corrector scheme which might be more efficient.
- For nonquadratic objective functions $f(x)$ one should add a line search, presumably a sufficient-decrease line search as in [7, section 11.5] and [10, sections 19.3, 19.4]. See also the modified back-tracking line searches in [1].
- For nonconvex objective functions $f(x)$ we could adjust the Hessian block in system (20) to always be positive definite so that the unconstrained primal step would be a descent direction. See [7, section 11.4].
- We could apply scalable linear algebra, presumably preconditioned Krylov methods [3], so that when matrices are sparse, as in Example 3, the solution of system (20) is faster.
- One could apply the quasi-Newton idea to construct an approximation of the Hessian block in system (20), so that the user does not need to supply a Hessian. See sections 12.3 and 13.5 of [7] and section 19.3 of [10].

References

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

[2] E. BUELER, *Stable finite volume element schemes for the shallow ice approximation*, J. Glaciol., 62 (2016), pp. 230–242.

[3] ——, *PETSc for Partial Differential Equations: Numerical Solutions in C and Python*, SIAM Press, Philadelphia, 2021.

[4] H. C. ELMAN, D. J. SILVESTER, AND A. J. WATHEN, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, 2nd ed., 2014.

[5] F. FACCHINEI AND J.-S. PANG, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Springer, 2007.

[6] M. GERTZ, J. NOCEDAL, AND A. SARTENAR, *A starting point strategy for nonlinear interior methods*, Applied Mathematics Letters, 17 (2004), pp. 945–952.

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

[8] D. KINDERLEHRER AND G. STAMPACCHIA, *An Introduction to Variational Inequalities and their Applications*, Academic Press, New York, 1980.

[9] R. J. LEVEQUE, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, SIAM Press, Philadelphia, 2007.

[10] J. NOCEDAL AND S. WRIGHT, *Numerical Optimization*, Springer, New York, 2nd ed., 2006.

[11] G. STRANG, *Karmarkar's algorithm and its place in applied mathematics*, The Mathematical Intelligencer, 9 (1987), pp. 4–10.

[12] L. N. TREFETHEN AND D. BAU, *Numerical Linear Algebra*, SIAM Press, Philadelphia, 1997.

[13] H. YAMASHITA AND H. YABE, *Superlinear and quadratic convergence of some primal-dual interior point methods for constrained optimization*, Mathematical Programming, 75 (1996), pp. 377–397.

[14] Y. ZHANG, R. A. TAPIA, AND J. E. DENNIS, JR, *On the superlinear and quadratic convergence of primal-dual interior point linear programming algorithms*, SIAM Journal on Optimization, 2 (1992), pp. 304–324.

Appendix: Invertibility of the system matrix

Recall from (14) that if $\mu_k > 0$ then x_k, λ_k are strictly positive. It follows that the diagonal matrices Λ_k, X_k in (16) are invertible.

Lemma. *Suppose $\nabla^2 f(x_k)$ is positive semi-definite. Then the $(n+m)\times(n+m)$ symmetric matrix in linear system (20) is invertible.*

Proof. Let $K = \nabla^2 f(x_k) + X_k^{-1}\Lambda_k$. It is easy to see that K is positive definite, thus invertible, since $X_k^{-1}\Lambda_k$ is diagonal with positive diagonal entries.

Suppose the system matrix has a null space:

$$\begin{bmatrix} K & -A^\top \\ -A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\tau \end{bmatrix} = 0. \tag{40}$$

The first row of (40) says

$$K\Delta x = A^\top \Delta\tau,$$

equivalently $\Delta x = K^{-1}A^\top\Delta\tau$. Now the second row of (40), $A\Delta x = 0$, can be written as an equation for $\Delta\tau$:

$$AK^{-1}A^\top\Delta\tau = 0.$$

Since K^{-1} is also positive definite, it has a Cholesky decomposition [12], so $K^{-1} = LL^\top$ where L is $n\times n$, lower triangular, and invertible. Recall that since A has full row rank its columns span $\mathbb{R}^m$, thus AL also has full row rank. But then $AK^{-1}A^\top = (AL)(AL)^\top$ is positive definite, thus invertible, so $\Delta\tau = 0$. Now $\Delta x = 0$, so the null space is trivial and the matrix is invertible. □

Corollary. *The system matrix in LP problem (32) is invertible.*

Proof. The zero matrix is positive semi-definite. □

Note that in (32) we have $K^{-1} = \Lambda_k^{-1}X_k = D^2$, with positive diagonal factors D, as the Cholesky decomposition. The quirky article by Strang [11] expands on the importance of matrices $AD^2A^\top$ where D is an invertible diagonal matrix and A has full row rank.