IPOPT Notes

Aditya Bhardwaj

These are my notes on the details of IPOPT. Currently, these notes assume familiarity with Newton's method, KKT conditions, and duality. Also, I have not yet typed up my notes on the problem scaling and initialization, the inertia correction for the KKT system, and the filter reset and watchdog accelerating heuristics used in the filter line search. Eventually, I hope to make these notes self-contained and better organized.

1 Problem Formulation

IPOPT solves the nonlinear optimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) && \text{(1a)} \\ \text{s.t.} \quad & g_L \le g(x) \le g_U && \text{(1b)} \\ & x_L \le x \le x_U && \text{(1c)} \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, and $g : \mathbb{R}^n \to \mathbb{R}^m$ are the constraints. The vectors g_L and g_U are the lower and upper bounds on the constraints, and x_L and x_U are the lower and upper bounds on the variables.

To understand the details of the algorithm we will consider the simpler case of

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) && \text{(2a)} \\ \text{s.t} \quad & c(x) = 0 && \text{(2b)} \\ & x \ge 0 && \text{(2c)} \end{aligned}$$

where $c : \mathbb{R}^n \to \mathbb{R}^m$ is the equality constraint and the vector inequality in (2c) holds component-wise. Note that in this problem formulation we can recover inequality constraints of the form $g(x) \ge 0$ for example by introducing the slack variable s and writing the constraint as $g(x) + s = 0$ with $s \ge 0$. Hence, if we modify the problem in (2) to include the constraint $x_L \le x \le x_U$ instead of just $x \ge 0$ we can recover the general problem in (1).

Notation 1.1. Let x^* denote the optimal solution to the problem in (2).

2 Algorithm Details

In this section, we describe the implementation details of IPOPT. First, we discuss how barrier methods work in general. Then we discuss how IPOPT works at a high level before zooming in on the details.

2.1 Barrier Method

First notice that we can rewrite the problem in (2) as

$$\min_{x \in \mathbb{R}^n} \quad f(x) + I_+(x) \tag{3a}$$

$$\text{s.t} \quad c(x) = 0 \tag{3b}$$

where the indicator function I_+ is defined as

$$I_+(x) = \begin{cases} 0 & \text{if } x \geq 0 \\ \infty & \text{otherwise.} \end{cases}$$

However, the problem in (3) is rather unpleasant since the indicator function is not differentiable. The idea of a barrier method is to replace the indicator function with a smooth function that approximates it. A popular choice is the negative logarithm function which is what IPOPT uses. Specifically, IPOPT solves the barrier problem

$$\min_{x \in \mathbb{R}^n} \quad \phi_\mu(x) \equiv f(x) - \mu \sum_{i=1}^{n} \log(x_i) \tag{4a}$$

$$\text{s.t} \quad c(x) = 0. \tag{4b}$$

Notation 2.1. Let $\ell(x) = -\sum_{i=1}^n \log(x_i)$ denote the logarithmic barrier function.

Note that as x_i approaches its bound of zero, $\log(x_i) \to -\infty$, which ensures that a solution to the barrier problem (4) will satisfy the bound constraint $x \geq 0$. The barrier parameter μ sets the tradeoff between minimizing the objective function and remaining in the feasible region. Additionally, the approximation to the indicator function becomes better and better for smaller values of μ. IPOPT (approximately) solves the barrier problem (4) for a fixed value of μ. Then it decreases μ and solves the barrier problem again using the solution from the previous problem. This continues for a sequence of barrier parameters decreasing to $\mu = 0$. As an interior-point method, IPOPT requires that the initial guess x_0 be feasible, i.e. it satisifies the bound constraint $x \geq 0$, but not necessarily the equality constraint $c(x) = 0$.

Intuitively, the barrier method seems like a promising way to solve the problem in (2). Let us prove that it actually works.

Definition 2.2. Let x_μ^* denote the solution to the barrier problem in (4) for a given μ. Such points are called *central points* and the set of all such points for all $\mu > 0$ is called the *central path.*

Proposition 2.3. *All x_μ^* are strictly feasible meaning that $x_\mu^* \geq 0$ and $c(x_\mu^*) = 0$. Additionally, for all x_μ^*, there exists a Lagrange multiplier $\lambda^* \in \mathbb{R}^m$ such that*

$$\begin{aligned} \nabla f(x_\mu^*) + \mu \nabla \ell(x_\mu^*) + \left(\frac{\partial c}{\partial x}\right)^\top \lambda^* &= \nabla f(x_\mu^*) - \mu \sum_{i=1}^n \frac{1}{x_{\mu,i}^*} e_i + \left(\frac{\partial c}{\partial x}\right)^\top \lambda^* \qquad (5) \\ &= 0. \end{aligned}$$

where e_i is an n-dimensional vector with a 1 in the i-th position and 0 elsewhere.

Proof. This is apparent from the limiting behavior of the logarithmic barrier function and the KKT conditions for the problem in (4). $\square$

Proposition 2.4. *Every central point x_μ^* gives rise to a dual feasible point and hence provides a lower bound on $f(x^*)$.*

Proof. Define $\nu_{\mu,i}^* = -\frac{\mu}{x_{\mu,i}^*}$ for $i = 1, \ldots, n$. Then ν_μ^*, λ^* are dual feasible. We show this as follows.

First, since $x_\mu^* \geq 0$, we have $\nu_\mu^* \leq 0$. Next, we can rewrite (5) as

$$\nabla f(x_\mu^*) + \nu_\mu^* + \left(\frac{\partial c}{\partial x}\right)^\top \lambda^* = 0$$

which means that x_μ^* minimizes the Lagrangian

$$L(x, \mu, \lambda) = f(x) + \nu^\top x + \lambda^\top c(x)$$

for $\nu = \nu_\mu^*$ and $\lambda = \lambda^*$. Hence, assuming that $f(x_\mu^*)$ is finite, we see that ν_μ^*, λ^* are dual feasible with the finite dual function

$$\begin{aligned} g(\nu_\mu^*, \lambda^*) = \inf_{x \in \mathbb{R}^n} L(x, \mu, \lambda) &= f(x_\mu^*) + (\nu_\mu^*)^\top x_\mu^* + (\lambda^*)^\top c(x_\mu^*) \qquad (6) \\ &= f(x_\mu^*) - \mu n. \end{aligned}$$

where the last equality follows from the fact that $c(x_\mu^*) = 0$ and $\nu_{\mu,i}^* = -\frac{\mu}{x_{\mu,i}^*}$. Since the dual function provides a lower bound on $f(x^*)$, it follows from (6) that

$$f(x_\mu^*) - f(x^*) \leq \mu n. \qquad (7)$$

Hence, we see that as we make μ smaller and smaller, x_μ^* converges to the optimal value x^*. $\square$

Given this result, a natural question to ask is that if we want to solve our problem to a certain tolerance, why don't we just make μ sufficiently small and solve (4) with Newton's method instead of a sequence of problems with decreasing barrier parameter μ? It turn out the reason is because this method of solving just the single problem does not work too well in practice. I think this is because the Hessian of ϕ_μ is ill-conditioned when you are close to the boundary.

The argument above convinces us that the barrier method works, but for the convergence proof of the actual IPOPT algorithm refer to [3].

2.2 Solving the Barrier Problem

In this section, we discuss how IPOPT solves the barrier problem in (4). First, let us consider the original problem in (2). The KKT conditions are

$$\nabla f(x) + \left(\frac{\partial c}{\partial x}\right)^\top \lambda - \nu = 0 \tag{8a}$$

$$c(x) = 0 \tag{8b}$$

$$x, \nu \geq 0 \tag{8c}$$

$$\nu_i x_i = 0 \quad \text{for} \quad i = 1, \dots n \tag{8d}$$

where $N \coloneqq diag(\nu)$, $X \coloneqq diag(x)$, and $\vec{1}$ is the vector of all ones. Now notice that by replacing the $\mu \sum_{i=1}^{n} \frac{1}{x_i} e_i$ term in (5) with ν, we can rewrite the optimality conditions in Prop. 2.3 for a central point x_μ^* as

$$\nabla f(x) + \left(\frac{\partial c}{\partial x}\right)^\top \lambda - \nu = 0 \tag{9a}$$

$$c(x) = 0 \tag{9b}$$

$$x, \nu \geq 0 \tag{9c}$$

$$\nu_i x_i = \mu \quad \text{for} \quad i = 1, \dots n. \tag{9d}$$

But this is almost exactly the same as the KKT conditions in (8) except for the last complementary slackness condition! As $\mu \to 0$ in (9), we recover (8). This is telling us that the central points almost satisfy the KKT conditions for the original problem and as we make μ smaller and smaller, the central points converge to the optimal solution x^*.

Notation 2.5. Let $N \coloneqq \text{diag}(\nu)$, $X \coloneqq \text{diag}(x)$, and $\vec{1}$ be the vector of all ones. Then we can rewrite (9d) as $NX\vec{1} - \mu\vec{1} = 0$, consistent with notation in the IPOPT papers.

IPOPT solves the barrier problem for a fixed value of the barrier parameter μ_j with a damped Newton's method by linearizing the KKT conditions in (9) around the current iterate (x_k, λ_k, ν_k) and solving the resulting linear system.

We denote the outer iterations where μ is updated with the index j and the inner iterations where the linear system is solved for a fixed μ_j with the index k. The linear system that gives the search directions is

$$\begin{pmatrix} H_k & \left(\frac{\partial c}{\partial x_k}\right)^\top & -I \\ \frac{\partial c}{\partial x_k} & 0 & 0 \\ N_k & 0 & X_k \end{pmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \\ d_k^\nu \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) + \left(\frac{\partial c}{\partial x_k}\right)^\top \lambda_k - \nu_k \\ c(x_k) \\ N_k X_k \vec{1} - \mu_j \vec{1} \end{pmatrix}. \tag{10}$$

where we use the sloppy notation $\frac{\partial c}{\partial x_k}$ to denote the Jacobian of c evaluated at x_k, and H_k is the Hessian of the Lagrangian $\nabla^2_{xx} L(x_k, \lambda_k, \nu_k)$ where L is the Lagrangian

$$L(x, \lambda, \nu) = f(x) + \lambda^\top c(x) - \nu. \tag{11}$$

Instead of directly solving (10) to get the search directions, we can be more efficient by instead first solving the smaller, symmetric linear system

$$\begin{pmatrix} H_k + \Sigma_k & \left(\frac{\partial c}{\partial x_k}\right)^\top \\ \frac{\partial c}{\partial x_k} & 0 \end{pmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla \phi_\mu(x_k) + \left(\frac{\partial c}{\partial x_k}\right)^\top \lambda_k \\ c(x_k) \end{pmatrix} \tag{12}$$

which we obtain through elimination with $\Sigma_k := X_k^{-1} N_k$. You can pass your favorite linear solver to IPOPT for solving this linear system (HSL MA57, MUMPS, Pardiso, etc.). The search direction for ν is then obtained via

$$d_k^\nu = \mu_j X_k^{-1} \vec{1} - X_k^{-1} d_k^x. \tag{13}$$

In practice, in order to ensure that we obtain a descent direction IPOPT instead solves the linear system

$$\begin{pmatrix} H_k + \Sigma_k + \delta_w I & \left(\frac{\partial c}{\partial x_k}\right)^\top \\ \frac{\partial c}{\partial x_k} & -\delta_c I \end{pmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla \phi_\mu(x_k) + \left(\frac{\partial c}{\partial x_k}\right)^\top \lambda_k \\ c(x_k) \end{pmatrix} \tag{14}$$

for $\delta_w, \delta_c \geq 0$. Recall that the nullspace of the constraint Jacobian $\frac{\partial c}{\partial x_k}$ gives us the feasible directions i.e. the directions that we can step in that locally keep us on the constraint manifold. Let Z_k be a matrix whose columns form a basis for the nullspace of $\frac{\partial c}{\partial x_k}$. Then in order to ensure that we obtain a descent direction, we require that the projection $Z_k^\top (H_k + \Sigma_k) Z_k$ of the top-left block of the matrix on the LHS of (12) onto the nullspace of $\frac{\partial c}{\partial x_k}$ is positive definite. This is just the standard Hessian regularization idea in Newton's method. The parameters δ_w and δ_c are chosen by a trial-and-error heuristic in IPOPT. The δ_c block is included to ensure the KKT matrix is nonsingular when the constraint Jacobian is rank-deficient.

After solving (14) and (13) for the search directions, we need to determine the step sizes α_k, α_k^ν to get the next iterate

$$x_{k+1} = x_k + \alpha_k d_k^x \tag{15a}$$

$$\lambda_{k+1} = \lambda_k + \alpha_k^\lambda d_k^\lambda \tag{15b}$$

$$\nu_{k+1} = \nu_k + \alpha_k^\nu d_k^\nu. \tag{15c}$$

IPOPT uses the same step size for the primal x and dual λ but a separate step size for the dual ν. In [4], they claim they use a separate step size for ν because they find in their experience that this is more efficient. Why do they not use a separate step size for λ then too? I think it is just because in general, the more step sizes you have, the trickier things become and you have to be careful that your primal and dual updates don't progress at significantly different rates. Also, equality constraints are easier to deal with than inequality constraints and so adding a separate step size for λ likely does not buy you that much.

How does IPOPT determine the step sizes α_k, α_k^ν? Since the barrier function blows up at $x = 0$, we have that that both the primal x and dual ν are positive at an optimal solution of the barrier problem. IPOPT maintains this positivity condition by using a *fraction-to-the-boundary* rule for the step sizes. For each outer iteration j, IPOPT maintains a fraction-to-the-boundary parameter

$$\tau_j = \max(\tau_{\min}, 1 - \mu_j) \tag{16}$$

for some $\tau_{min} \in (0, 1)$. Then the step size for the dual ν is given by

$$\alpha_k^\nu = \max\{\alpha \in (0, 1] : \nu_k + \alpha d_k^\nu \geq (1 - \tau_j)\nu_k\} \tag{17}$$

and the maximum step size for the primal x and dual λ is given by

$$\alpha_k^{\max} = \max\{\alpha \in (0, 1] : x_k + \alpha d_k^x \geq (1 - \tau_j)x_k\}. \tag{18}$$

When μ_j is small $1 - \tau_j$ is only slightly bigger than 0, so the fraction-to-the-boundary rule ensures that the step sizes are never big enough to make x, ν negative i.e infeasible. Note that (18) only gives us the maximum step size. To determine the actual step size, IPOPT uses a backtracking line search where it keeps decreasing $\alpha_{\max}$ by a factor of 2 until it finds a satisfactory step size. We now describe the filter line search method that IPOPT uses to determine how to find a satisfactory step size.

Notation 2.6. Let $\alpha_{k,l}$ denote the value of the step size at the l-th iteration of the line search. Namely, $\alpha_{k,l} = 2^{-l}\alpha_k^{\max}$.

2.3 Filter Line Search

The filter line search is based on the idea that we can view the problem as trying to optimize two things: minimizing the objective and minimizing the constraint violation $\theta(x) := ||c(x)||_1$. With these considerations, IPOPT considers a step size $\alpha_{k,l}$ to be acceptable if $x_k(\alpha_{k,l}) := x_k + \alpha_{k,l} d_k^x$ satisfies one of the following conditions:

$$\theta(x_k(\alpha_{k,l})) \leq (1 - \gamma_\theta)\theta(x_k) \tag{19a}$$

$$\phi_{\mu_j}(x_k(\alpha_{k,l})) \leq \phi_{\mu_j}(x_k) - \gamma_\phi\theta(x_k) \tag{19b}$$

The condition (19a) says we accept if the constraint violation decreases by more than some factor γ_θ of the current constraint violation and the condition (19b) says that the step needs to decrease the objective function value by more than some factor γ_ϕ of the constraint violation. Both γ_ϕ and γ_θ are fixed constants in $(0, 1)$. By default, IPOPT uses the values $\gamma_\phi = 10^{-8}$ and $\gamma_\theta = 10^{-5}$. In the case, where the constraint violation is small enough, IPOPT instead checks the following “switching” conditions

$$\nabla\phi_{\mu_j}(x_k)^\top d_k^x < 0 \quad \text{and} \quad \alpha_{k,l}(-\nabla\phi_{\mu_j}(x_k)^\top d_k^x)^{s_\phi} > \delta(\theta(x_k))^{s_\theta} \tag{20}$$

with constants $\delta, s_\theta > 1$ and $s_\phi \geq 1$. The first condition is just making sure that the search direction actually leads to a decrease according to the gradient and the second condition is making sure that this expected decrease from the gradient is bigger than some power of the constraint violation modulo constants. This seems like a reasonable thing, but I am not really sure what motivates this heuristic or how they came up with it.

If (20) is satisfied and the constraint violation is small enough $\theta(x_k) \leq \theta^{\min}$, then the Armijo condition

$$\phi_{\mu_j}(x_k(\alpha_{k,l})) \leq \phi_{\mu_j}(x_k) + \eta_\phi \alpha_{k,l} \nabla\phi_{\mu_j}(x_k)^\top d_k^x \tag{21}$$

needs to be satisfied in order for the step to be acceptable. The parameter η_ϕ must be in $(0, \frac{1}{2})$ and its default value is 10^{-8}. The default value for $\theta_{\min}$ is 0.0001.

So what is the filter part of the line search then? The filter $\mathcal{F}_k$ is a set of (θ, ϕ) pairs that keeps track of the objective function value and constraint violation value pairs that are prohibited from being accepted. That is, a trial point $x(\alpha_{k,l})$ is rejected if the pair $(\theta(x(\alpha_{k,l})), \phi_{\mu_j}(x(\alpha_{k,l})))$ is in the filter $\mathcal{F}_k$. For each barrier problem, the filter is initialized as

$$\mathcal{F}_0 = \{(\theta, \phi) \in \mathbb{R}^2 : \theta \geq \theta^{\max}\} \tag{22}$$

so we never accept points that have constraint violation bigger than $\theta^{\max}$. At each iteration, if the Armijo condition (21) is not satisfied, or if we are not in the “switching” mode where we check (20) and $\theta(x_k) \leq \theta_{\min}$, then the filter is updated as

$$\mathcal{F}_{k+1} = \mathcal{F}_k \cup \{(\theta, \phi) \in \mathbb{R}^2 : \theta \geq (1 - \gamma_\theta)\theta(x_k) \quad \text{and} \quad \phi \geq \phi_{\mu_j}(x_k) - \gamma_\phi \theta(x_k).\} \tag{23}$$

which are the same as the conditions in (19). This makes sure that we never end up cycling and returning to a neighborhood of a previously accepted point x_p for $p = k, \ldots, 1$. Otherwise, if we are in the “switching” mode and the Armijo condition is satsifed then the filter is not updated.

Question 2.7. *What if we decrease the step size by a lot and still don’t find a step that satisfies one of the conditions above?*

IPOPT computes a minimum allowable step size $\alpha_k^{\min}$. If we end up decreasing our step size to the point where $\alpha_{k_l} \leq \alpha_k^{\min}$, then IPOPT switches to what it calls a *feasiblity restoration phase* where it tries to find a new feasible point $x_{k+1} \geq 0$ that passes the tests in (19) and the resulting updated filter $\mathcal{F}_{k+1}$. It accomplishes this by solving a smooth reformulation of the optimization problem

$$\begin{aligned} \min_{\bar{x}\in\mathbb{R}^n} \quad & ||c(\bar{x})||_1 + \frac{\zeta}{2}||D_R(\bar{x} - x_k)||_2^2 && \text{(24a)} \\ \text{s.t.} \quad & \bar{x} \geq 0 && \text{(24b)} \end{aligned}$$

where $\zeta > 0$ is a penalty parameter, the second term in the objective is included to penalize deviating too much from x_k, the point where the restoration phase was entered, and

$$D_R = \operatorname{diag}\left(\min\left(1, \frac{1}{|x_{k,1}|}\right), \dots, \min\left(1, \frac{1}{|x_{k,n}|}\right)\right). \tag{25}$$

This choice of D_R is to ensure the second term in the objective term in (24) does not blow up if x_k is small. The smooth reformulation of (24) that IPOPT actually solves is given by rewriting (24) in the form of (2):

$$\begin{aligned} \min_{\bar{x}\in\mathbb{R}^n, p, n\in\mathbb{R}^m} \quad & \sum_{i=1}^{m}(p_i + n_i) + \frac{\zeta}{2}||D_R(\bar{x} - x_k)||_2^2 && \text{(26a)} \\ \text{s.t.} \quad & c(\bar{x}) - p + n = 0 && \text{(26b)} \\ & \bar{x}, p, n \geq 0. && \text{(26c)} \end{aligned}$$

Here p and n correspond to positive and negative parts of the constraint respectively. Now, we can just solve this restoration phase problem using the exact same IPOPT algorithm we have been discussing! We solve (26) until we obtain an iterate $\bar{x}_t$ that is acceptable to the filter $\mathcal{F}_{k+1}$ for our actual problem and satisfes $\theta(\bar{x}_t) \leq \kappa_{\text{resto}}\theta(x_k)$ where $\kappa_{\text{resto}} \in (0, 1)$ is a constant with default value 0.9. The reason for this second condition is to ensure the restoration phase makes significant progress towards feasibliity. Since the restoration phase only terminates for $\bar{x}_t$ that satisfies the filter $\mathcal{F}_{k+1}$, we already know that $\bar{x}_t$ lowers the infeasibility and the objective function value. So this second condition is IPOPT trying to have its cake and get a much bigger slice! (The fraction $1 - 0.9 = 0.1$ is much bigger than the default value of γ_θ.) It turns out this works well in practice according to [4]. Finally, if the restoration phase fails to find a point that satisfies the filter, then it should converge to a stationary point of the constraint violation, indicating that the problem is at least locally infeasible. Initializing the solver with a different initial guess can help in this case.

We have discussed the filter line search, but it turns out IPOPT actually tries to avoid this if possible by using a second order correction method.

2.4 Second Order Correction

The second order correction method is used when the first step size in the line search $\alpha_{k,0}$ fails to be acceptable and leads to no improvement or a bigger constraint violation $\theta(x_(\alpha_{k,0})) \geq \theta(x_k)$ than the current iterate. Since in the first step we already computed $c(x_k + \alpha_{k_0} d_k^x)$ we obtain the second order correction direction $d_k^{x,\mathrm{soc}}$ by solving

$$\left(\frac{\partial c}{\partial x_k}\right)^\top d_k^{x,\mathrm{soc}} + c(x_k + \alpha_{k,0} d_k^x) = 0. \tag{27}$$

This uses the gradient at the old point x_k and the constraint function evaluated at the faiiled trial point $x_k + \alpha_{k,0} d_k^x$. Hence, it is just asking for a step direction that makes the first order Taylor approximation of the constraint function at $x_k + \alpha_{k,0} d_k^x$ vanish where the Jacobian $\frac{\partial c}{\partial x_k}$ is used proxy for the true Jacobian at $x_k + \alpha_{k,0} d_k^x$. The corrected search direction is then given by

$$d_k^{x,\mathrm{cor}} = \alpha_{k,0} d_k^x + d_k^{x,\mathrm{soc}}. \tag{28}$$

The easiest way to understand this is by trying to visualize what this is doing by drawing the standard Newton's method picture for this case. You will see how it gets you closer to the zero.

Once the corrected search direction is obtained, the same fraction-to-the-boundary rule as in (17) is used to determine the step size α_k^{soc} for the corrected search direction. Then if $x_k^{\mathrm{soc}} = x_k + \alpha_k^{\mathrm{soc}} d_k^{x,\mathrm{cor}}$ satisfies the filter and sufficient decrease conditions—either (19) or if the constraint violation is small enough (20) and (21)—then the next iterate x_{k+1} is given by x_k^{soc}. Otherwise, IPOPT keeps computing additional second order corrections until it either finds a point that satisfies the filter and sufficient decrease conditions or terminates after reaching the maximum number of second order corrections or finding a correction that does not decrease the constraint violation by a constant factor $\kappa_{\mathrm{soc}} \in (0,1)$ with default value 0.99. In the latter case, IPOPT then reverts back to the standard backtracking filter line search outlined in the previous section.

One final major detail that we have not yet discussed is how IPOPT updates the barrier parameter μ and determines when to stop. After solving the barrier problem for μ_j, the next barrier parameter is given by

$$\mu_{j+1} = \max\left(\frac{\epsilon_{\mathrm{tol}}}{10}, \min\left(\kappa_\mu \mu_j, \mu_j^{\theta_\mu}\right)\right) \tag{29}$$

with $\kappa_\mu \in (0,1)$ and $\theta_\mu \in (1,2)$ with default values of 0.2 and 1.5 respectively. The update rule in (29) ensures that the barrier parameter is eventually decreased superlinearly when we get to small enough values of μ, but also no more than the desired tolerance to which we wish to solve our problem (recall (7) and the discussion afterwards). Finally, each barrier problem terminates when the KKT conditions in (9) are satisfied to within some tolerance modulo some scaling factors and the entire IPOPT algorithm terminates when the KKT conditions in (8) are satisfied to within some tolerance again modulo scaling factors.

To summarize, IPOPT works by solving a sequence of barrier problems with decreasing barrier parameter. The solution for each previous barrier problem is used as the initial guess for the next barrier problem. Each barrier problem is solved using a damped Newton's method with search directions for each iteration obtained by solving the regularized, inertia-corrected KKT system in (14). The step sizes are determined by a backtracking filter line search that defaults to the second order correction method if the first step size in the line search fails to be acceptable. If the step size is decreased to a certain minimum allowable step size and an acceptable next iterate still hasn't been found, then IPOPT switches to the feasiblity restoration phase. The barrier parameter is updated according to (29) and the algorithm terminates when the KKT conditions are satisfied to within some tolerance.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, March 2004.

[2] Robert M. Freund. Penalty and barrier methods for constarined optimization, 2004. [Accessed 07-01-2024].

[3] Andreas Wächter and Lorenz T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005.

[4] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, April 2005.