Interior-Point and Augmented Lagrangian Algorithms for Optimization and Control

Stephen Wright

University of Wisconsin-Madison

May 2014
My first talk was about optimization formulations, optimality conditions, and duality — for LP, QP, LCP, and nonlinear optimization.

This section will review some algorithms, in particular:

- Primal-dual interior-point (PDIP) methods
- Augmented Lagrangian (AL) methods.

Both are useful in control applications. We’ll say something about PDIP methods for model-predictive control, and how they exploit the structure in that problem.
Recapping Gradient Methods

Considering unconstrained minimization

$$\min f(x)$$

where $f$ is smooth and convex, or the constrained version in which $x$ is restricted to the set $\Omega$, usually closed and convex.

First-order or gradient methods take steps of the form

$$x_{k+1} = x_k - \alpha_k g_k,$$

where $\alpha_k \in \mathbb{R}_+$ is a steplength and $g_k$ is a search direction. $d_k$ is constructed from knowledge of the gradient $\nabla f(x)$ at the current iterate $x = x_k$ and possibly previous iterates $x_{k-1}, x_{k-2}, \ldots$.

Can extend to nonsmooth $f$ by using the subgradient $\partial f(x)$.

Extend to constrained minimization by projecting the search line onto the convex set $\Omega$, or (similarly) minimizing a linear approximation to $f$ over $\Omega$. 

Wright (UW-Madison)
Can view the gradient method step $x_{k+1} = x_k - \alpha_k g_k$ as the minimization of a first-order model of $f$ plus a “prox-term” which prevents the step from being too long:

$$x_{k+1} = \arg \min_x f(x_k) + g_k^T (x - x_k) + \frac{1}{2\alpha_k} \|x - x_k\|_2^2.$$ 

Taking the gradient the quadratic and setting to zero, we obtain

$$g_k + \frac{1}{\alpha_k} (x_{k+1} - x_k) = 0,$$

which gives the formula for $x_{k+1}$.

This viewpoint is the key to several extensions.
Extensions: Constraints

When a constraint set $\Omega$ is present we can simply minimize the quadratic model function over $\Omega$:

$$x_{k+1} = \arg \min_{x \in \Omega} f(x_k) + g_k^T(x - x_k) + \frac{1}{2\alpha_k} \|x - x_k\|_2^2.$$ 

Gradient Projection has this form.

We can replace the $\ell_2$-norm measure of distance with some other measure $\phi(x; x_k)$:

$$x_{k+1} = \arg \min_{x \in \Omega} f(x_k) + g_k^T(x - x_k) + \frac{1}{2\alpha_k} \phi(x; x_k).$$

Could choose $\phi$ to “match” $\Omega$. For example, a measure derived from the entropy function is a good match for the simplex

$$\Omega := \{x \mid x \geq 0, \ e^T x = 1\}.$$
Extensions: Regularizers

In many modern applications of optimization, \( f \) has the form

\[
f(x) = l(x) + \tau \psi(x).
\]

Can extend the prox approach above by

- choosing \( g_k \) to contain gradient information from \( l(x) \) only;
- including \( \tau \psi(x) \) explicitly in the subproblem.

Subproblems are thus:

\[
x_{k+1} = \arg\min_x l(x_k) + g_k^T (x - x_k) + \frac{1}{2\alpha_k} \|x - x_k\|^2 + \tau \psi(x).
\]
Rather than penalizing distance moved from current $x_k$, we can enforce an explicit constraint: a trust region.

$$x_{k+1} = \arg \min_x f(x_k) + g_k^T (x - x_k) + I_{\|x-x_k\| \leq \Delta_k}(x),$$

where $I_{\Lambda}(x)$ denotes an indicator function with

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

Adjust trust-region radius $\Delta_k$ to ensure progress e.g. descent in $f$. 

Constrained Optimization
May 2014 7 / 46
Could use the original $f$ in the subproblem rather than a simpler model function:

$$x_{k+1} = \underset{x}{\text{arg min}} \; f(x) + \frac{1}{2\alpha_k} \|x - x_k\|_2^2.$$  

although the subproblem seems “just as hard” to solve as the original, the prox-term may make it easier by introducing strong convexity, and may stabilize progress.

Can extend to constrained and regularized cases also.
Quadratic Models: Newton’s Method

We can extend the iterative strategy further by adding a quadratic term to the model, instead of (or in addition to) the simple prox-term above.

**Taylor’s Theorem** suggests basing this term on the Hessian (second-derivative) matrix. That is, obtain the step from

\[
x_{k+1} := \arg \min_x f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2} (x - x_k)^T \nabla^2 f(x_k) (x - x_k).
\]

Can reformulate to solve for the step \(d_k\): Then have \(x_{k+1} = x_k + d_k\), where

\[
d_k := \arg \min_d f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d.
\]

See immediately that this model won’t have a bounded solution if \(\nabla^2 f(x_k)\) is not positive definite. It usually *is* positive definite near a strict local solution \(x^*\), but need something that works more globally.
One “obvious” strategy is to add the prox-term to the quadratic model:

\[ d_k := \arg \min_d f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \left( \nabla^2 f(x_k) + \frac{1}{\alpha_k} I \right) d, \]

choosing \( \alpha_k \) so that

- The quadratic term is positive definite;
- Some other desirable property holds, e.g. descent \( f(x_k + d_k) < f(x_k) \).

We can also impose the trust-region explicitly:

\[ d_k := \arg \min_d f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d + I_{\|d\| \leq \Delta_k} (x), \]

or alternatively:

\[ d_k := \arg \min_{d : \|d\| \leq \Delta_k} f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d. \]

But this is equivalent: For any \( \Delta_k \), there exists \( \alpha_k \) such that the solutions of the prox form and the trust-region form are identical.
Another disadvantage of Newton is that the Hessian may be difficult to evaluate or otherwise work with. The quadratic model is still useful when we use first-order information to learn about the Hessian.

Key observation (from Taylor’s theorem): the secant condition:

$\nabla^2 f(x_k)(x_{k+1} - x_k) \approx \nabla f(x_{k+1}) - \nabla f(x_k)$.

The difference of gradients tells us how the Hessian behaves along the direction $x_{k+1} - x_k$. By aggregating such information over multiple steps, we can build up an approximation to the Hessian than is valid along multiple directions.

Quasi-Newton Methods maintain an approximation $B_k$ to $\nabla^2 f(x_k)$ that respects the secant condition.

The approximation may be implicit rather than explicit, and we may store an approximation to the inverse Hessian instead.
A particularly popular quasi-Newton method, suitable for large-scale problems is the limited-memory BFGS method (L-BFGS) which stores the Hessian or inverse Hessian approximation implicitly.

L-BFGS stores the last $5 - 10$ update pairs:

$$s_j := x_{j+1} - x_j, \quad y_j := \nabla f(x_{j+1}) - \nabla f(x_j).$$

for $j = k, k - 1, k - 2, \ldots, k - m$. Can implicitly construct $H_{k+1}$ that satisfies

$$H_{k+1}y_j = s_j.$$

In fact, an efficient recursive formula is available for evaluating $d_{k+1} := -H_{k+1}\nabla f(x_{k+1})$ — the next search direction — directly from the $(s_j, y_j)$ pairs and from some initial estimate of the form $(1/\alpha_{k+1})I$. 

There is also a variant of Newton’s method for nonlinear equations: Find $x$ such that

$$F(x) = 0,$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ (n equations in n unknowns.)

Newton’s method forms a linear approximation to this system, based on another variant of Taylor’s Theorem which says

$$F(x + d) = F(x) + J(x)d + \int_0^1 [J(x + td) - J(x)]d \, dt,$$

where $J(x)$ is the Jacobian matrix of first partial derivatives:

$$J(x) = \begin{bmatrix}
\frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n}
\end{bmatrix} \quad \text{(usually not symmetric).}$$
When $F$ is continuously differentiable, we have

$$F(x_k + d) \approx F(x_k) + J(x_k)d,$$

so the Newton step is the one that make the right-hand side zero:

$$d_k := -J(x_k)^{-1}F(x_k).$$

The basis Newton method takes steps $x_{k+1} := x_k + d_k$. Its effectiveness can be improved by

- Doing a line search: $x_{k+1} := x_k + \alpha_k d_k$, for some $\alpha_k > 0$;
- Levenberg strategy: add $\lambda I$ to $J$ and set $d_k := -(J(x_k) + \lambda I)^{-1}F(x_k)$.

Guide progress via a merit function, usually

$$\phi(x) := \frac{1}{2} \|F(x)\|^2_2.$$

Achtung! Can get stuck in a local min of $\phi$ that’s not a solution of $F(x) = 0$. 

Homotopy

Tries to avoid the local-min issue with the merit function. Start with an “easier” set of nonlinear equations, and gradually deform it to the system \( F(x) = 0 \), tracking changes to the solution as you go.

\[
F(x, \lambda) := \lambda F(x) + (1 - \lambda)F_0(x), \quad \lambda \in [0, 1].
\]

Assume that \( F(x, 0) = F_0(x) = 0 \) has solution \( x_0 \). Homotopy methods trace the curve of solutions \((x, \lambda)\) until \( \lambda = 1 \) is reached. The corresponding value of \( x \) then solves the original problem.

- Many variants.
- Some supporting theory.
- Typically more expensive than enhanced Newton methods, but better at finding solutions to \( F(x) = 0 \).

We mention homotopy mostly because of its connection to interior-point methods.
Recall the monotone LCP: Find \( z \in \mathbb{R}^n \) such that

\[
0 \leq z \perp Mz + q \geq 0,
\]

where \( M \in \mathbb{R}^{n \times n} \) is positive semidefinite, and \( q \in \mathbb{R}^n \). Recall too that monotone LCP is a generalization of LP and convex QP.

Rewrite LCP as

\[
w = Mz + q, \quad (w, z) \geq 0, \quad w_i z_i = 0, \quad i = 1, 2, \ldots, n,
\]

which is a constrained system of nonlinear equations:

\[
F(z, w) = 0, \quad (z, w) \geq 0,
\]

where \( F : \mathbb{R}^{2n} \rightarrow \mathbb{R}^{2n} \) is defined as

\[
F(z, w) = \begin{bmatrix}
Mz + q - w \\
WZe
\end{bmatrix} = \begin{bmatrix}
0 \\
0
\end{bmatrix},
\]

where

\[
W = \text{diag}(w_1, w_2, \ldots, w_n), \quad Z = \text{diag}(z_1, z_2, \ldots, z_n), \quad e = (1, 1, \ldots, 1)^T.
\]
Interior-Point Methods

Interior-point methods generate iterates that satisfy the nonnegativity constraints strictly, that is,

\[(z^k, w^k) > 0 \quad \text{for all } k.\]

An obvious strategy is to search along Newton directions for \( F \), with a line search to maintain positivity of \((z^{k+1}, w^{k+1})\). The Newton equations are:

\[
\begin{bmatrix}
M & -I \\
W & Z
\end{bmatrix}
\begin{bmatrix}
\Delta z \\
\Delta w
\end{bmatrix}
= -
\begin{bmatrix}
Mz + q - w \\
WZe
\end{bmatrix}
\]

This affine scaling approach can actually work, and convergence can be proved, but it needs very precise conditions on the steplength to avoid getting jammed at the boundary.

Much better performance is obtained from methods that use homotopy, in particular, the idea of following a central path.
The central path is a critical object in primal-dual interior-point methods. It’s defined by the parametric equations:

\[
F(z, w; \tau) := \begin{bmatrix}
Mz + q - w \\
WZe - \tau e
\end{bmatrix} = \begin{bmatrix}
0 \\
0
\end{bmatrix},
\]

where \( \tau \geq 0 \) is the central path parameter, along with the conditions \( (z, w) > 0 \).

The second block of conditions is saying that

\[
w_i z_i = \tau, \quad \text{for all } i = 1, 2, \ldots, n,
\]

so if \( \tau > 0 \), we must have all components of \( w \) and \( z \) strictly positive — hence the term interior point.\(^1\)

\(^1\)Actually, interior-point methods don’t force the first condition \( Mz + q - w = 0 \) to hold except in the limit, so the iterates are not really “interior” in the sense of being “strictly feasible.”
The central path guides iterates to the solution. The basic primal-dual interior-point method works as follows:

- Compute Newton steps for $F(z, w; \tau_k)$ for some value of $\tau_k$.
- Take a step
  $$(z^{k+1}, w^{k+1}) = (z^k, w^k) + \alpha_k(\Delta z^k, \Delta w^k),$$
  choosing $\alpha_k$ so that $(z^{k+1}, w^{k+1})$ remains sufficiently positive.
- Possibly enhance with some auxiliary “corrector” steps.
- Choose a smaller value $\tau_{k+1} < \tau_k$ and repeat.

The effect is that it chases a moving target along the central path, staying reasonably close to the central path to avoid getting jammed near the boundary of the region $(w, z) \geq 0$. 

Wright (UW-Madison)
A Simple PDIP Method

Usually works, simple to code (one page!)

- Choose $\tau_{k+1} = .9\tau_k$;
- Choose $\alpha_k$ to be the largest $\alpha$ for which

$$
(z_i^k + \alpha \Delta z_i^k)(w_i^k + \alpha \Delta w_i^k) \geq .01(z^k + \alpha \Delta z^k)^T(w^k + \alpha \Delta w^k)/n.
$$

The second condition ensures that iterates stay in a loose neighborhood of the central path — none of the variables go to zero prematurely.
A More Elaborate PDIP Method (Mehrotra)

Chooses $\tau_{k+1}$ by a heuristic based on the performance of the affine scaling step $(\Delta z^k_{\text{aff}}, \Delta w^k_{\text{aff}})$, which is computed by solving the Newton equations with $\tau = 0$.

- If we can take a long step along the affine scaling direction before reaching the boundary, choose $\tau_{k+1}$ much smaller than $\tau_k$ — an aggressive choice.
- Otherwise, if the affine scaling step quickly jams at the boundary, make a more conservative choice — $\tau_{k+1}$ only a little smaller than $\tau_k$.

Mehrotra’s method also includes a corrector step, that corrects for the nonlinearity revealed by the affine-scaling step. The combined centering-corrector step is obtained from

$$
\begin{bmatrix}
M & -I \\
W & Z
\end{bmatrix}
\begin{bmatrix}
\Delta z_{cc} \\
\Delta w_{cc}
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
\tau_{k+1}e - \Delta z^k_{\text{aff}}\Delta w^k_{\text{aff}}e
\end{bmatrix}.
$$

This is added to the affine-scaling step to obtain the search direction.
Solving the Linear Equations

At each iteration need to solve two or more linear systems of the form:

\[
\begin{bmatrix}
M & -I \\
W & Z
\end{bmatrix}
\begin{bmatrix}
\Delta z \\
\Delta w
\end{bmatrix}
= \begin{bmatrix}
r_f \\
r_{zw}
\end{bmatrix},
\]

where

- \(W\) and \(Z\) are the diagonals of \(w^k\) and \(z^k\) which contain all positive numbers.
- \(r_f\) and \(r_{zw}\) are different right-hand sides (affine-scaling, centering-corrector).

Note that there is a lot of structure in this system — three of the blocks are \(N \times N\) nonsingular diagonal — and that the structure is the same at every interior-point iteration.
Reduced Form

Can do block elimination (of $\Delta w$) to obtain

$$(M + Z^{-1}W)\Delta z = r_f + Z^{-1}r_{wz}.$$ 

Note that

- When $M$ is positive semidefinite (as it is for monotone LCP, then $M + Z^{-1}W$ is positive definite).
- If we can identify a good strategy for reordering rows and columns of $M$ to solve this efficiently, we can re-use this ordering at every iteration, because $M + Z^{-1}W$ has the same nonzero pattern (only the diagonals change).
- Since some elements of $Z^{-1}W$ are going to $\infty$ as the iterates near a solution while others are going to zero, this system can become highly ill-conditioned — but it doesn’t seem to matter much.
When we apply this technique to LCPs arising from LP, \( M \) has a particular form, and the linear system that we solve has the form:

\[
\begin{bmatrix}
0 & A^T & I \\
A & 0 & 0 \\
S & 0 & X
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \lambda \\
\Delta s
\end{bmatrix}
= 
\begin{bmatrix}
rp \\
r_d \\
r_{xs}
\end{bmatrix},
\]

where \( X = \text{diag}(x_1, x_2, \ldots, x_n) \) and \( S = \text{diag}(s_1, s_2, \ldots, s_n) \). By eliminating \( \Delta s \) we obtain a reduced form:

\[
\begin{bmatrix}
-S^{-1}X & A^T \\
A & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \lambda
\end{bmatrix}
= 
\begin{bmatrix}
r_p - S^{-1}r_{xs} \\
r_d
\end{bmatrix}.
\]

Since \( S^{-1}X \) is positive diagonal we can go a step further and eliminate \( \Delta x \):

\[
A(SX^{-1})A^T \Delta \lambda = r_d + ASX^{-1}(r_p - S^{-1}r_{xs}).
\]
Thus we have a coefficient matrix of the form $ADA^T$ where

- $D = SX^{-1}$ is positive diagonal.
- Some elements of $D$ go to zero as the iterates converge (those for which $s_i = 0$ at the solution) while others go to $\infty$ (those for which $x_i = 0$). Hence the system MAY be ill conditioned.
- In practical codes, this matrix product is actually formed and factored at every iteration, using a variant of the Cholesky factorization.
- This factorization is stable regardless of the ordering of rows/columns, so we are free to use orderings that create the least fill-in during Cholesky.

Good codes: MOSEK, Gurobi, CPLEX. Freebie: PCx.
Given current state $x_0$, choose a time horizon $N$ (long), and solve the optimization problem for $x = \{x_k\}_{k=0}^{N}$, $u = \{u_k\}_{k=0}^{N-1}$:

$$
\min L(x, u), \quad \text{subject to } x_0 \text{ given,}
\quad x_{k+1} = F_k(x_k, u_k), \quad k = 0, 1, \ldots, N - 1, 
\quad \text{other constraints on } x, u.
$$

Then apply controls $u_0, u_1, u_2, \ldots$ blindly, without monitoring the state.

- flexible with respect to nonlinearity and constraints;
- if model $F_k$ is inaccurate, solution may be bad;
- doesn't account for system disturbances during time horizon;
- never used in industrial practice!
A canonical control problem. For given $x_0$, solve:

$$\min_{x, u} \Phi(x, u) := \frac{1}{2} \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \quad \text{s.t.} \quad x_{k+1} = A x_k + B u_k.$$ 

From KKT conditions, dependence of optimal values of $x_1, x_2, \ldots$ and $u_0, u_1, \ldots$ on initial $x_0$ is linear. We can substitute these variables out to obtain

$$\min_{x, u} \Phi(x, u) = \frac{1}{2} x_0^T \Pi x_0,$$

for some s.p.d. matrix $\Pi$.

By using this dynamic programming principle, isolating the first stage, can write the problem as:

$$\min_{x_1, u_0} \frac{1}{2} (x_0^T Q x_0 + u_0^T R u_0) + \frac{1}{2} x_1^T \Pi x_1 \quad \text{s.t.} \quad x_1 = A x_0 + B u_0.$$
By substituting for $x_1$, get unconstrained quadratic problem in $u_0$. Minimizer is

$$u_0 = Kx_0, \quad \text{where} \quad K = -(R + B^T \Pi B)^{-1}B^T \Pi A.$$ 

so that

$$x_1 = Ax_0 + Bu_0 = (A + BK)x_0.$$ 

By substituting for $u_0$ and $x_1$ in

$$\frac{1}{2}x_0^T \Pi x_0 = \frac{1}{2}(x_0^T Q x_0 + u_0^T R u_0) + \frac{1}{2}x_1^T \Pi x_1,$$

obtain the Riccati equation:

$$\Pi = Q + A^T \Pi A - A^T \Pi B(R + B^T \Pi B)^{-1}B^T \Pi A.$$ 

There are well-known techniques to solve this equation for $\Pi$, hence $K$.

Hence, we have a feedback control law $u = Kx$ that is optimal for the LQR problem. This is a closed-loop control strategy that can respond to changes in state.
Closed-loop has the highly desirable property of being able to respond to system disturbances and modeling errors. But closed-form solutions can be found only for simple models such as LQR.

Open-loop allows for a rich variety of models and constraints, and yields a highly structured optimization problem that can often be solved efficiently. But it can’t fix disturbances and errors. “Set and forget” at your peril!

Model-Predictive Control (MPC) is a way of doing closed-loop control using open-loop / optimal-control techniques.

- Set up a optimal control problem, initialized at the current state, with a finite planning horizon.
- Solve this open-loop problem (quickly!) and implement the control $u_0$.
- At the next decision point, repeat the process. Can use the previous solution (shifted forward one period) as a warm start.
A generalization of the linear-quadratic regulator (LQR) problem includes constraints:

$$\min_{x,u} \frac{1}{2} \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k,$$

subject to

$$x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, 2, \ldots,$$
$$x_k \in X, \quad u_k \in U,$$

possibly also mixed constraints, and constraints on $u_{k+1} - u_k$.

Assuming that $0 \in \text{int}(X)$, $0 \in \text{int}(U)$ and that the system is stabilizable, we expect that $u_k \to 0$ and $x_k \to 0$ as $k \to \infty$. Therefore, for large enough $k$, the non-model constraints become inactive.
Hence, for $N$ large enough, the problem is equivalent to the following (finite) problem:

$$
\min_{x,u} \frac{1}{2} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k + \frac{1}{2} x_N^T \Pi x_N,
$$

subject to

$$
x_{k+1} = A x_k + B u_k, \quad k = 0, 1, 2, \ldots, N - 1
$$

$$
x_k \in X, \quad u_k \in U, \quad k = 0, 1, 2, \ldots, N - 1,
$$

where $\Pi$ is the solution of the Riccati equation. In the “tail” of the sequence ($k > N$) simply apply the unconstrained LQR feedback law, derived above.

When constraints are linear, need to solve a finite, structured, convex quadratic program.
Interior-Point Method for MPC

\[ \min_{u,x,\epsilon} \sum_{k=0}^{N-1} \frac{1}{2} (x_k^T Q x_k + u_k^T R u_k + 2x_k^T M u_k + \epsilon_k^T Z \epsilon_k) + z^T \epsilon_k + x_N^T \Pi x_N, \]

subject to

- \( x_0 = \hat{x}_j \), (fixed)
- \( x_{k+1} = A x_k + B u_k, \) \( k = 0, 1, \ldots, N - 1, \)
- \( D u_k - G x_k \leq d, \) \( k = 0, 1, \ldots, N - 1, \)
- \( H x_k - \epsilon_k \leq h, \) \( k = 1, 2, \ldots, N, \)
- \( \epsilon_k \geq 0, \) \( k = 1, 2, \ldots, N, \)
- \( F x_N = 0. \)

- Soft constraints on the state (involving \( \epsilon_k \));
- Also general “hard” constraints on \((x_k, u_k)\). Can use these to implement constraints on control changes \( u_{k+1} - u_k \).
Introduce dual variables, use stagewise ordering. Primal-dual interior-point method yields a block-banded system at each iteration:

\[
\begin{bmatrix}
\ldots & Q & M & -G^T & A^T \\
M^T & R & D^T & B^T \\
-G & D & -\Sigma^D_k & \\
A & B & -\Sigma^\epsilon_{k+1} & -l \\
-l & -\Sigma^H_{k+1} & -l & H \\
-l & -l & Z & \\
-l & H^T & Q & \ldots
\end{bmatrix}
\begin{bmatrix}
\Delta x_k \\
\Delta u_k \\
\Delta \lambda_k \\
\Delta p_{k+1} \\
\Delta \xi_{k+1} \\
\Delta \eta_{k+1} \\
\Delta \epsilon_{k+1} \\
\Delta x_{k+1} \\
\vdots
\end{bmatrix}
= \begin{bmatrix}
\vdots \\
r_k^\times \\
r_k^u \\
r_k^\lambda \\
r_k^p \\
r_k^\xi \\
r_k^\eta \\
r_k^\epsilon \\
r_{k+1}^\times \\
\vdots
\end{bmatrix}
\]

where \( \Sigma^D_k, \Sigma^\epsilon_{k+1} \), etc are diagonal.
By performing block elimination, get reduced system

\[
\begin{bmatrix}
R_0 & B^T \\
B & -I \\
-1 & Q_1 & M_1 & A^T \\
M_1^T & R_1 & B^T \\
A & B & -I \\
-1 & Q_2 & M_2 & A^T \\
M_2^T & R_2 & B^T \\
A & B & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots \\
\cdots & Q_N & F^T \\
F & F^T \\
\end{bmatrix}
\begin{bmatrix}
\Delta u_0 \\
\Delta p_0 \\
\Delta x_1 \\
\Delta u_1 \\
\Delta p_1 \\
\Delta x_2 \\
\Delta u_2 \\
\vdots \\
\Delta x_N \\
\Delta \beta \\
\end{bmatrix}
= \begin{bmatrix}
\tilde{r}_0^u \\
\tilde{r}_0^p \\
\tilde{r}_1^x \\
\tilde{r}_1^u \\
\tilde{r}_1^p \\
\tilde{r}_2^x \\
\tilde{r}_2^u \\
\vdots \\
\tilde{r}_N^x \\
r_\beta \\
\end{bmatrix}.
\]

which has the same structure as the KKT system of a problem without side constraints (soft or hard).
Can solve by applying a banded linear solver: $O(N)$ operations. Alternatively, seek matrices $\Pi_k$ and vectors $\pi_k$ such that the following relationship is satisfied between $\hat{\Delta}p_{k-1}$ and $\hat{\Delta}x_k$:

$$-\hat{\Delta}p_{k-1} + \Pi_k \hat{\Delta}x_k = \pi_k, \quad qaudk = N, N - 1, \ldots, 1.$$ 

By substituting in the linear system, find a recurrence relation:

$$\Pi_N = \bar{Q}_N, \quad \pi_N = \tilde{r}^x_N,$$ 

$$\Pi_{k-1} = Q_{k-1} + A^T \Pi_k A -$$

$$(A^T \Pi_k B + M_{k-1})(R_{k-1} + B^T \Pi_k B)^{-1}(B^T \Pi_k A + M_{k-1}^T),$$

$$\pi_{k-1} = \tilde{r}^x_{k-1} + A^T \Pi_k \tilde{r}^p_{k-1} + A^T \pi_{k-1} -$$

$$(A^T \Pi_k B + M_{k-1})(R_{k-1} + B^T \Pi_k B)^{-1}(\tilde{r}^u_{k-1} + B^T \Pi_k \tilde{r}^p_{k-1} + B^T \pi_{k-1}).$$

The recurrence for $\Pi_k$ is the discrete time-varying Riccati equation!
Recall yesterday’s conversation about duality for general constrained problems:

$$\min f(x) \text{ subject to } c_i(x) \geq 0, \quad i = 1, 2, \ldots, m.$$ 

with the Lagrangian defined by:

$$\mathcal{L}(x, \lambda) := f(x) - \lambda^T c(x) = f(x) - \sum_{i=1}^{m} \lambda_i c_i(x).$$

Remember that we defined primal and dual objectives:

$$r(x) = \sup_{\lambda \geq 0} \mathcal{L}(x, \lambda);$$

$$q(\lambda) = \inf_{x} \mathcal{L}(x, \lambda),$$

so the primal and dual problems are:

$$\min_x r(x), \quad \max_{\lambda \geq 0} q(\lambda).$$
Augmented Lagrangian

Can motivate it crudely as proximal point on the dual. Consider the dual:

$$\max_{\lambda \geq 0} q(\lambda) = \max_{\lambda \geq 0} \inf_x f(x) - \lambda^T c(x).$$

Add a proximal point term — a quadratic penalty on the distance moved from the last dual iterate $\lambda^k$:

$$\max_{\lambda \geq 0} \inf_x f(x) - \lambda^T c(x) - \frac{1}{2\alpha_k} \|\lambda - \lambda^k\|^2_2$$

Note that the objective here is “simple” in $\lambda$. Now switch the max and inf:

$$\inf_x \left\{ \max_{\lambda \geq 0} f(x) - \lambda^T c(x) - \frac{1}{2\alpha_k} \|\lambda - \lambda^k\|^2_2 \right\}.$$

We can solve the max problem, to get an explicit value for $\lambda$! This is easy because the components of $\lambda$ are separated in the objective; we can solve for each one individually.
Explicit solution is

$$\lambda_i = \begin{cases} 0 & \text{if } -\alpha_k c_i(x) + \lambda_i^k \leq 0; \\ -\alpha_k c_i(x) + \lambda_i^k & \text{otherwise.} \end{cases}$$

Substitute this result in the previous expression to get

$$\min_x f(x) + \sum_{i : c_i(x) \geq \lambda_i^k / \alpha_k} \frac{1}{2\alpha_k} (\lambda_i^k)^2$$

$$+ \sum_{i : c_i(x) < \lambda_i^k / \alpha_k} \left( \frac{\alpha_k}{2} c_i(x)^2 - \lambda_i^k c_i(x) \right).$$

Thus the basic augmented Lagrangian process is (iteration $k$):

- Minimize the augmented Lagrangian function above for $x$ (approximately); call it $x_{k+1}$;
- Plug this $x$ into the explicit-max formula for $\lambda$ to get $\lambda^{k+1}$;
- Choose $\alpha_{k+1}$ for the next iteration.
Equality Constraints

For the equality constrained case, the formulae simplify a lot. We have

\[ \min_x f(x) \text{ subject to } d_j(x) = 0, \ j = 1, 2, \ldots, p. \]

Dual objective is \( \inf_x \mathcal{L}(x, \mu) := f(x) - \mu^T d(x) \).

Augmented Lagrangian subproblems are:

\[
\begin{align*}
    x^{k+1} &= \arg \min_x f(x) - (\mu^k)^T d(x) + \frac{\alpha_k}{2} \|d(x)\|_2^2, \\
    \mu^{k+1} &= \mu^k - \alpha_k d(x^{k+1}).
\end{align*}
\]
Augmented Lagrangian: History and Practice

- How accurately to solve the subproblem for $x$, which is unconstrained but nonlinear?
- How to adjust $\alpha_k$? Use a different $\alpha_k$ value for each constraint; increase it when this constraint is not becoming feasible rapidly enough.

Historical Sketch:

- Developments in 1970s- early 1980s by Rockafellar, Bertsekas, others.
- Largely lost favor as an approach for general nonlinear programming during the next 15 years.
- Recent revival in the context of sparse optimization and its many applications, in conjunction with splitting / coordinate descent.
Separable Objectives: ADMM

Alternating Directions Method of Multipliers (ADMM) arises when the objective in the basic linearly constrained problem is separable:

$$\min_{(x,z)} f(x) + h(z) \text{ subject to } Ax + Bz = c,$$

for which

$$\mathcal{L}(x, z, \lambda; \rho) := f(x) + h(z) - \lambda^T (Ax + Bz - c) + \frac{\alpha}{2} \|Ax - Bz - c\|_2^2.$$

Standard augmented Lagrangian would minimize $\mathcal{L}(x, z, \lambda; \rho)$ over $(x, z)$ jointly — but these are coupled through the quadratic term, so the advantage of separability is lost.

Instead, minimize over $x$ and $z$ separately and sequentially:

$$x_k = \arg \min_x \mathcal{L}(x, z_{k-1}, \lambda_{k-1}; \alpha_k);$$

$$z_k = \arg \min_z \mathcal{L}(x_k, z, \lambda_{k-1}; \alpha_k);$$

$$\lambda_k = \lambda_{k-1} - \alpha_k (Ax_k + Bz_k - c).$$
Basically, does a round of block-coordinate descent in \((x, z)\).

The minimizations over \(x\) and \(z\) add only a quadratic term to \(f\) and \(h\), respectively. This does not alter the cost much.

Can perform these minimizations inexactly.

Convergence is often slow, but sufficient for many applications.

Many recent applications to compressed sensing, image processing, matrix completion, sparse principal components analysis.

ADMM has a rich collection of antecedents. A nice recent survey, including a diverse collection of machine learning applications, is Boyd et al. (2011).
ADMM for Consensus Optimization

Given

\[
\min \sum_{i=1}^{m} f_i(x),
\]

form \(m\) copies of the \(x\), with the original \(x\) as a "master" variable:

\[
\min_{x, x^1, x^2, \ldots, x^m} \sum_{i=1}^{m} f_i(x^i) \text{ subject to } x^i = x, \ i = 1, 2, \ldots, m.
\]

Apply ADMM, with \(z = (x^1, x^2, \ldots, x^m)\), get

\[
x_k^i = \arg \min_{x^i} f_i(x^i) - (\lambda_{k-1}^i)^T (x^i - x_{k-1}) + \frac{\alpha_k}{2} \|x^i - x_{k-1}\|_2^2, \forall i,
\]

\[
x_k = \frac{1}{m} \sum_{i=1}^{m} \left( x_k^i - \frac{1}{\alpha_k} \lambda_{k-1}^i \right),
\]

\[
\lambda_k^i = \lambda_{k-1}^i - \alpha_k (x_k^i - x_k), \forall i
\]

Obvious parallel possibilities in the \(x^i\) updates. Synchronize for \(x\) update.
The feasible set is sometimes an intersection of two or more convex sets that are easy to handle separately (e.g. projections are easily computable), but whose intersection is more difficult to work with.

**Example:** Optimization over the cone of doubly nonnegative matrices:

\[
\min_{X} f(X) \text{ s.t. } X \succeq 0, \ X \geq 0.
\]

General form:

\[
\min_{x} f(x) \text{ s.t. } x \in \Omega_i, \ i = 1, 2, \ldots, m
\]

Just consensus optimization, with indicator functions for the sets.

\[
x_k = \arg \min_{x} f(x) - \sum_{i=1}^{m} (\lambda_{k-1}^i)^T (x - x_{k-1}^i) + \frac{\alpha_k}{2} \|x - x_{k-1}^i\|_2^2,
\]

\[
x_k^i = \arg \min_{x_i \in \Omega_i} -(\lambda_{k-1}^i)^T (x_k - x^i) + \frac{\alpha_k}{2} \|x_k - x^i\|_2^2, \ \forall i
\]

\[
\lambda_k^i = \lambda_{k-1}^i - \alpha_k (x_k - x_k^i), \ \forall i.
\]
ADMM and Prox-Linear

Given

$$\min_x f(x) + \tau \psi(x),$$

reformulate as the equality constrained problem:

$$\min_{x,z} f(x) + \tau \psi(z) \text{ subject to } x = z.$$

ADMM form:

$$x_k := \min_x f(x) + \tau \psi(z_{k-1}) + (\lambda_k)^T (x - z_{k-1}) + \frac{\alpha_k}{2} \|z_{k-1} - x\|_2^2,$$

$$z_k := \min_z f(x_k) + \tau \psi(z) + (\lambda_k)^T (x_k - z) + \frac{\alpha_k}{2} \|z - x_k\|_2^2,$$

$$\lambda_{k+1} := \lambda_k + \alpha_k(x_k - z_k).$$

- Minimization over $z$ is the shrink operator — often inexpensive.
- Minimization over $x$ can be performed approximately using an algorithm suited to the form of $f$. 
The subproblems are not too different from those obtained in prox-linear algorithms:

- $\lambda_k$ is asymptotically similar to the gradient term in prox-linear, that is, $\lambda_k \approx \nabla f(x_k)$;
- Thus, the minimization over $z$ is quite similar to the prox-linear step.


