Final Report on:

Improving Performance of the Interior Point Method by Preconditioning

-- *Application to Distributed Command and Control* --

Project by: Ken Ryals

For: AMSC 663-664

Fall 2007-Spring 2008

Contact Information:

e-mail: KenRyals "at" aol.com

Problem Sponsors:

Christopher C. Wright (JHU/APL)

Peter P. Pandolfini (JHU/APL)

Abstract

The Office of the Secretary of Defense/Acquisition and Technology (OSD/A&T), has a need for an optimization tool to use in their Distributed Command and Control System for nuclear assets. Several factors combine to imply that an Interior Point Method (IPM) for optimization would be applicable as it can easily address conic problems and it maintains iterate feasibility once a feasible point has been attained. The research presented herein addresses the stability of the Interior Point Method. In addition to using the standard techniques of factorization and iterative solution, the normal equations for the IPM will be preconditioned using an inverse obtained from the constraint matrix (specifically the inverse of $\boldsymbol{A}\,\boldsymbol{A}^{\boldsymbol{T}}$) to reduce the condition number for ill-conditioned problems. The proposed preconditioner was tested on several benchmark datasets in the NETLIB as a surrogate for the (classified) OSD/A&T dataset. Although the proposed preconditioner improved the stability on some of the test problems, it did not compare favorably with either the factorization or iterative solution techniques in either an average or majority sense. Thus, the added expense is not recommended as a general approach to improving performance of the IPM.

20 December 2007

Final Report on:
Improving Performance of the Interior Point Method by Preconditioning
-- Application to Distributed Command and Control --

Introduction

The Office of the Secretary of Defense/Acquisition and Technology (OSD/A&T), has a need for an optimization tool to use in their Distributed Command and Control System for nuclear assets. Although the problem is presently couched in a linear format, there is a potential for the constraint set to become more complex in the future, transitioning the optimization problem from a linear optimization to a second order conic optimization. Furthermore, the criticality of the application mandates that the optimization system be robust with respect to variations in the values used therein. Finally, changes in the system state should generally be in a subset of the full dimension of the problem, thereby suggesting that subsequent solutions will usually be similar. These factors combine to imply that an Interior Point Method for optimization would be applicable as it can easily address conic problems and it maintains iterate feasibility once a feasible point has been attained. The research proposed herein is intended to address the stability of the Interior Point Method in situations where the problem is ill-conditioned.

Background

A general linear optimization problem can be expressed as:

$$\min_{x} c^T x$$
$$\textit{subject to}:$$
$$Ax = b$$
$$x \geq 0$$

The dual to this problem is:

$$\max_{y} b^T y$$
$$\textit{subject to}:$$
$$A^T y + z = c$$
$$z \geq 0$$

where the original problem being has changed from one using $\boldsymbol{x}$ (the "*primal*" variable) to one using $\boldsymbol{y}$ (the "*dual*" variable) and $\boldsymbol{z}$ (the "*slack*" variable), and the optimization is changed from minimization to maximization. We define the "duality gap" as $\boldsymbol{\mu = x^T z}$. If $\boldsymbol{\mu = 0}$, both problems yield the same solution. Numerically, this pair of problems can be reduced to solving a linear system of equations:

$$\begin{bmatrix} Z & 0 & X \\ A & 0 & 0 \\ 0 & A^T & I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} \mu e - Xz \\ b - Ax \\ c - A^T y - z \end{bmatrix}$$

$$\textit{where } X \textit{ refers to } \mathrm{diag}(x) \textit{ and } e \textit{ is a vector of ones } (\mathbf{1}'s)$$

This system of equations is usually reduced to a system for the dual variable (**y**) of the form:

$$AD^{-2}A^T\Delta y = b$$

$$\textit{where } D^2 = X^{-1}Z\textit{, which is a positive definite diagonal matrix}$$

Thus, as **μ ➔0**, a family of solutions, ***(x,y,z)***, can be generated that approaches the constraint set from within the feasible region. (Although not germane to this research, details on the development of the "dual" problem and the reduced system of equations are presented in Addendum A for the interested reader.)

Motivation

In solving the reduced system of equations, the condition number of the matrix AD^2A^T increasing as one approaches the constraint set. Consider the following canonical problem:

$$\textbf{min } (-x_1 - 2x_2)$$

subject to the constraints:

$$\begin{aligned} -2x_1 + x_2 + x_3 &= 2 \\ -x_1 + 2x_2 + x_4 &= 7 \\ x_1 + 2x_2 + x_5 &= 3 \\ x_1; x_2; x_3; x_4; x_5 &\geq 0 \end{aligned}$$

which in matrix form is:

$$c^T = \begin{bmatrix} -1 & -2 & 0 & 0 & 0 \end{bmatrix} \quad A = \begin{bmatrix} -2 & 1 & 1 & 0 & 0 \\ -1 & 2 & 0 & 1 & 0 \\ 1 & 2 & 0 & 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 2 \\ 7 \\ 3 \end{bmatrix}$$

and the closed form solution is:

$$\begin{aligned} 0 \leq x_2 \leq 1\tfrac{1}{2} \quad & x_1 = 3 - 2x_2 \\ x_5 = 0 \quad & x_3 = 8 - 5x_2 \\ & x_4 = 10 - 4x_2 \end{aligned}$$

The dual form of this problem is:

$$\max_y (2y_1 + 7y_2 + 3y_3)$$

***subject to the constraints*:**

$$\begin{aligned} -2y_1 - y_2 + y_3 + z_1 &= -1 \\ y_1 + 2y_2 + 2y_3 + z_2 &= -2 \\ y_1 + z_3 &= 0 \\ y_2 + z_4 &= 0 \\ y_3 + z_5 &= 0 \\ z_1, z_2, z_3 &\geq 0 \end{aligned}$$

For which one can show the solution to be:

$$y = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T \quad \textit{and} \quad z = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

The optimal value (maximum) produced by this dual problem is "***-3***", the exact same optimum (minimum) value produced by the original (or primal) problem. Note that $\boldsymbol{x^T z = 0}$ at the combined optima. The product $\boldsymbol{x^T z}$ is referred to as the "duality gap" and is denoted as μ.

The set of plots below (Figures 1 and 2) depict the condition number for $\boldsymbol{AD^2A^T}$ and $\boldsymbol{D^2}$ as the parameter $\boldsymbol{\mu}$ decreases on successive iterations. Comparing the two plots, it is clear that the condition number for $\boldsymbol{AD^2A^T}$ does not continue to grow as the condition number for $\boldsymbol{D^2}$ does.

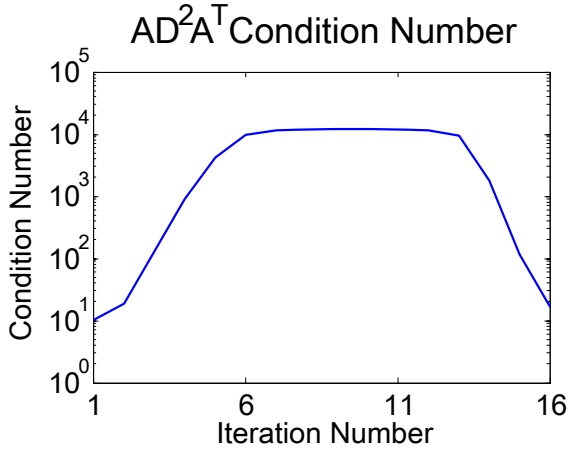

Figure 2 Condition Number History for AD^2A^T

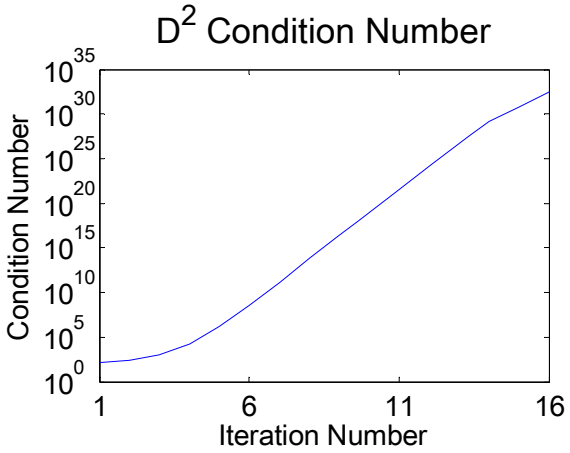

Figure 1 Condition Number History for D2

Next, multiply the first term in the constraint matrix ($\boldsymbol{A}$) by $\mathbf{10^7}$ to increase the condition number of $\boldsymbol{AA^T}$ from **11** to ≈**3.88 * 10^{14}**. The condition number plot for this ill-conditioned system is shown in Figure 3 and it exhibits the same behavior as the well conditioned system. This indicates that the $\boldsymbol{A}$ and $\boldsymbol{A^T}$ terms act to ameliorate the tendency of $\boldsymbol{D^2}$ to become ill-conditioned as iterations approach the constraint set.

Since $\boldsymbol{A}$ and $\boldsymbol{A^T}$ appear to act to ameliorate the tendency of $\boldsymbol{D^2}$ to become ill-conditioned in forming $\boldsymbol{AD^2A^T}$, it is proposed that pre-multiplying the reduced set of equations by $\boldsymbol{(AA^T)^{-1}}$ will improve the numerical behavior of the IPM as iterations approach the constraint set by reducing the condition number. The choice of $\boldsymbol{(AA^T)^{-1}}$ instead of $\boldsymbol{(AA^T)}$ for preconditioning is motivated by the following relationship in the case where the constraint matrix ($\boldsymbol{A}$) is square (and invertible):

$$\mathbf{(AA^T)^{-1}\ A\ D^2\ A^T = (A^T)^{-1}\ D^2\ A^T}$$

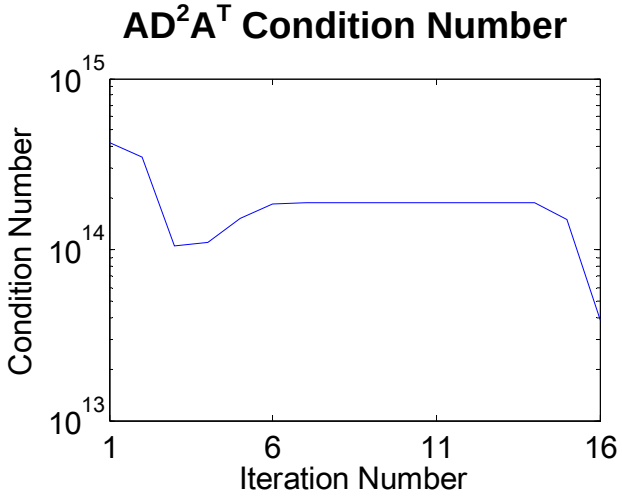

Figure 3 (Ill-)Condition Number History for AD^2A^T

In this case, $(AA^T)^{-1}AD^2A^T$ is equivalent to $(A^T)^{-1}D^2A^T$, which is a similarity transform on D^2, suggesting that $(AA^T)^{-1}AD^2A^T$ may be well behaved. Applying this preconditioning (by $(AA^T)^{-1}$) to the ill-conditioned test problem shown above produces the condition number plot shown in Figure 4 as the iterations increase and the solution approaches the constraint set. The $(AA^T)^{-1}AD^2A^T$ condition number ranges between approximately eight and two hundred, which is significantly smaller than the non-conditioned value for AD^2A^T (which is greater than $3*10^{13}$). Based on this positive result, it was deemed to be worthwhile to pursue the use of $(AA^T)^{-1}$ as a preconditioning matrix for the linear system of equations that must be solved in generating the IPM iterates.

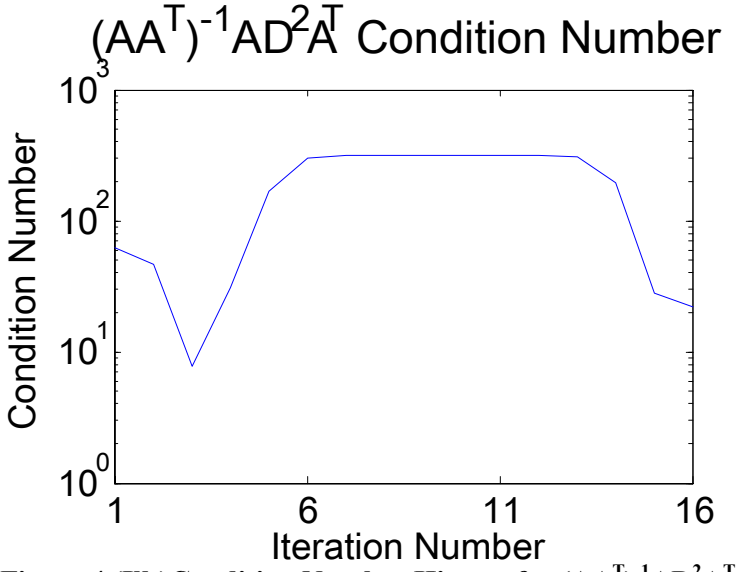

Figure 4 (Ill-)Condition Number History for $(AA^T)^{-1}AD^2A^T$

The final issue in using the proposed preconditioned IPM is in the method used for generating the solution of the linear system. The most prevalent approach is to factor the AD^2A^T matrix by Cholesky factorization and then solve the simpler sets of equations (by backward/forward substitution). Factorization has the benefit of halving the power of the condition number (for example, from 10^{14} to 10^7). In cases where the power becomes very large, this can be insufficient, hence, the search for an alternate approach. Nonetheless, the benefits of this approach are not to be ignored as a possible adjunct to the (new) approach being investigated. An alternate approach to directly solving the system of equations is to solve them iteratively by the conjugate gradient method. The conjugate gradient method frequently makes use of a preconditioner, which makes it compatible with the concept of using preconditioning in the IPM. Since the IPM solves a succession of similar problems, each solution should be in the neighborhood of the previous one; thus, a starting point for the preconditioned conjugate gradient (PCG) method is readily available. The rationale for the factorization and iterative solution techniques are detailed in the following paragraphs.

Benefit of Factorization

The general problem we are solving is of the form:

$\mathbf{AD^2A^T\,\Delta y} = \square$ (*where* □ *stands for something that is not important at the moment*)

where $\mathbf{D^2 \sim diag(x/z)}$

Re-writing this by splitting $\mathbf{D^2}$ in half ($\mathbf{D^2=D^TD}$) produces:

$\mathbf{AD^2A^T\,\Delta y = ADD^TA^T\,\Delta y} = \square$

Next, construct the QR factorization of $\mathbf{D^TA^T}$ so that:

$$\mathbf{D^TA^T=QR}$$

*w*here **Q** is an **n** by **n** and orthonormal matrix and **R** is an **n** by **m** upper triangular matrix

Substituting for $\mathbf{ADD^TA^T}$ gives:

$$[R\ 0]Q^T\ Q\begin{bmatrix} R \\ 0 \end{bmatrix}\Delta y$$

where the rectangular matrix **R** is represented by its square non-zero upper part ("**R**") and its zero lower part "**0**".

After multiplying this out and dropping the zero parts, we get:

$$\mathbf{R^TQ^TQR\ \Delta y} = \square$$

Since $\mathbf{Q^TQ = I}$ from the orthonormal nature of **Q**, we have

$$\mathbf{R^TR\ \Delta y} = \square$$

This equation can be solved for $\mathbf{\Delta y}$ simply using forward and backwards substitution. By this approach, the Cholesky Decomposition ($\mathbf{R^TR}$) of $\mathbf{AD^2A^T}$ is obtained without using $\mathbf{AD^2A^T}$ but "only half of it" ($\mathbf{D^TA^T}$) for much better stability.

Benefits of Iterative Solution

The Interior Point Method solves a series of problems that are successively closer approximations to the constrained problem being solved. Since the purpose of each solution is to provide an initial value to use in the next iteration of the IPM, the earlier solutions need not be exactly determined. Thus, approximate solutions to the linear system of equations in the IPM iterations can be used to improve computational efficiency. The conjugate gradient method is one very powerful iterative solution technique that is particularly suited to preconditioning. Thus, it will be examined for its impact on stability by linking the CG solution tolerance to the duality gap of the IPM. Specifically, the CG solver tolerance and the IPM duality gap were each halved for successive steps. This permitted the IPM to require only one iteration in each step to achieve the desired duality gap, once it had exited the initialization stage.

Performance Metrics

In order to compare the performance of these three algorithmic components on the stability of the IPM, a set of performance metrics are required. Since stability is the primary focus of this research, it will be assessed in two ways. First, the condition number of the matrix being solved for the dual increment ($\mathbf{\Delta y}$) at the solution will be used to used as a measure of the numerical difficulty involved in solving the linear system of equations. Then the "smoothness" of the three solution variables (**x**, **y**, and **z**) will be examined to determine the extent to which the solution undergoes rapid oscillations or switches between alternate solution sets. This will be measured

by the infinity-norm of the successive increments of these variables. This will collect the sum of each of the absolute values of the changes between iterations. Although stability is the primary concern, information on the number of iterations required to achieve a specified solution tolerance and the execution time required will be collected to assure that a stable but extremely inefficient algorithm is not selected. Relative to the execution time metric, it should be remembered that the conjugate gradient algorithm was designed to enhance visibility of its intermediate operations and is not as efficient as commercially available CG routines. Likewise, the initialization routine was specifically designed to produce valid, but inaccurate starting points so that the algorithms would be forced to work harder.

Development Process

The development environment for this project revolved around Matlab, so that C++ would be available for areas where it could confer significant speed improvements. The use of Matlab also permitted concurrent Verification and Validation (V&V) by use of built-in Matlab solution routines to generate solutions by alternate means at each stage of the process. This also allowed for risk mitigation by providing an alternate path to produce intermediate computational results within the overall architecture of the Preconditioned IPM (PIPM) system. Thus, the goal of evaluating the efficacy of this scheme was virtually assured, independent of software development issues. Combining the benefits of the development environment with a staged development permitted several assessment points throughout the generation of the desired PIPM system.

The main components developed in support of this investigation are:

- An IPM module
- A module to develop the factorization
    - The QR factorization of $\boldsymbol{DA^T}$ was used to generate the Cholesky factorization of $\boldsymbol{AD^2A^T}$.
- A module for the (preconditioned) Conjugate Gradient solution
    - A module was developed instead of using a commercially available routine to permit better access to intermediate computations.
- An initial value generation module
    - This module was structured to generate valid initial values that were not optimal so that the solution algorithms would not have trivial problems
- A module to load data and set up the problem
- A module to compute or collect statistics:
    1. Execution time
    2. Number of iterations required
    3. Final Condition Number

4. Solution variability throughout the iteration process.

- And a main routine to manage execution of the eight combinations of algorithms
  1. Basic IPM (no preconditioning, no factorization, no conjugate solver)
  2. Basic IPM plus preconditioning
  3. Basic IPM plus factorization
  4. Basic IPM plus preconditioning and factorization
  5. CG solver IPM (no preconditioning, no factorization)
  6. CG solver IPM plus preconditioning
  7. CG solver IPM plus factorization
  8. CG solver IPM plus preconditioning and factorization

Testing/V&V Plan

In addition to concurrent testing using the canonical problem shown and the "AFIRO" linear programming problem obtained from Dr. D. P. O'Leary during AMSC 607 (Advanced Numerical Optimization), the "AFIRO" problem and additional problems from the NETLIB LP Test Problems library of a comparable size to the OSD/A&T problem (30-80 variables) will be used to evaluate the software system (The NETLIB library is available at: http://www-fp.mcs.anl.gov/otc/GUIDE/TestProblems/LPtest/index.html).

Test Data Description

Obtaining the OSD A&T data, even in a sanitized form, was not possible within the time-frame of this project due to delays in the parent OSD A&T project resulting from the delay in Congress passing a DoD appropriations bill. Thus, as a risk Mitigation Strategy, several NETLIB LP test problems of appropriate dimension were identified to use as testing surrogates. These problems are shown in Table 1.

Table 1 Description of Selected NETLIB Linear Optimization Test Problems

Name	Rows	Cols	Non-zeros	Optimal Value
AFIRO	28	32	88	-4.65E+02
SC50A	51	48	131	-6.46E+01
SC50B	51	48	119	-7.00E+01
ADLITTLE	57	97	465	2.25E+05
BLEND	75	83	521	-3.08E+01

As a representative example of the nature of these problems, the sparsity pattern of the AFIRO problem is shown in Figure 5 (depicting 102 non-zeros, 27 constraint equations and 51 parameters).

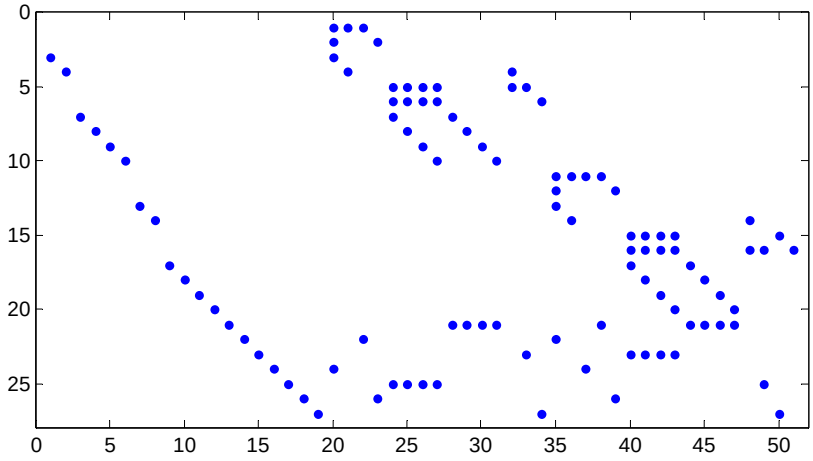

Figure 5 Sparsity Pattern for AFIRO Problem Constraint Matrix

Results

The data obtained from executing the eight algorithmic combinations on the test problems is shown in Addendum B. These results were examined in two different manners:

1. To determine "on average" what algorithmic components improved solution performance,
2. To determine what algorithmic components improved solution performance "most often".

The first comparison might emphasize larger, more complex problems over small, quickly solved problems. Thus, the two examinations as a pair should reveal both the likelihood of an algorithmic component improving performance and the degree to which it tends to improve performance.

To examine what algorithmic components improved solution performance "on average", a generalized linear model for each of the four performance metrics was generated. The parameters of these models is shown in the table below:

Variable	intercept	if CG	if Precond	if QR
Iterations	101.2	44.7	24.8	-4.7
Execution_time	0.0	11.8	2.5	1.0
Log(Condition)	17.1	-4.0	3.8	-5.4
Log(x_variability)	2.8	-0.1	-0.1	0.1
Log(y_variability)	9.7	-8.9	2.6	-3.0
Log(z_variability)	9.7	-8.5	2.6	-3.0
Legend		Significant Improvement	Significant Deterioration	Not significant

A GLM fits a multi-dimensional "line" through the data; for example:

$$Iterations \approx 101.2 + 44.7*(if\_CG) + 24.8*(if\_Precond) - 4.7*(if\_QR)$$

Thus, using the conjugate gradient solver (if CG) increases the number of iterations by approximately 44.7, using preconditioning increases the number of iterations by approximately 24.8, and using the QR factorization decreases the number of iterations by approximately 4.7.

These results indicate that using the QR factorization is beneficial for all metrics but execution time, the CG iterative solution to the linear system of equations improves the stability metrics, and the preconditioning did not improve any metrics. The increase in execution time when using the QR factorization is a result of the time necessary to compute the factorization. For the problems examined, this amounted to only one second on average. Generating approximate solutions to intermediate problems by the conjugate gradient algorithm slowed the solution process (in terms of time and iterations) but improved the stability metrics by decreasing the changes between successive values of the solution variables (**x**,**y**,**z**).

To examine which algorithmic components improved solution performance the "most often", the algorithms (i.e., particular combination of components) were ranked from best (#1) to worst (#8) and those rankings were averaged over the five problems. The results of these rankings are shown in the table below for each of the eight algorithms:

Algorithm		Iteration Rank	Execution Time Rank	Condition Rank	x Variation Rank	y Variation Rank	z Variation Rank
0	Basic	2.6	2.2	4.8	5.6	6.6	6.6
1	Preconditionning	2	2.4	8	5.2	7	7
2	QR	3	3.6	3	4	5.8	5.8
3	Pre and QR	3.2	4.8	6	5	6.2	6.2
4	CG	5	4.2	4.2	4	2.2	1.8
5	PCG	6.4	6	5.8	4.2	2.8	2.4
6	CG and QR	5	5.4	1.4	3.4	2.2	2
7	PCG and QR	6	7	2.8	3.6	2.2	2.2
Legend		Significant Deterioration			Significant Improvement		

Combining results over the different algorithms by algorithmic component produces the results shown below:

Algorithm	Iteration Rank	Execution Time Rank	Condition Rank	x Variation Rank	y Variation Rank	z Variation Rank
Preconditionning	4.4	5.05	5.65	4.5	4.55	4.45
QR	4.3	5.2	3.3	4	4.1	4.05
CG	5.6	5.65	3.55	3.8	2.35	2.1

Either way of examining the ranking data reveals the fact that preconditioning may have helped occasionally, but it is not generally beneficial to either solution speed or stability. On the other hand, using the conjugate gradient solver was always beneficial to solution stability (at some added cost), and the QR factorization generally improved solution stability (overall condition number and x variable stability) at some added cost.

The improvement to stability afforded by using the CG method to iteratively solve the linear systems of equations is something of a surprise. Apparently, the iteratively obtained intermediate estimates produce a smoother path to the final solution and provide a more stable algorithm. The use of factorization was expected to improve solution stability and it did for the major two areas – Condition Number and **x** (primary variable) variability.

Summary

The recommended algorithm is to use the QR factorization with the CG solver. This permits one to gradually lower the CG tolerance for solution of the system of linear equations while the IPM optimization algorithm decreases the duality gap. For general usage, one would want to improve the initial value generator and use one of the standard techniques for initializing IPM problems in place of the intentionally "bad" initialization used herein. In re-optimizing the Distributed Command and Control network, the past solution should provide a better initial point, since changes should generally be on a minor nature. Finally, it must be noted that the proposed preconditionner did not uniformly improve performance. It showed promise on some problems, so it is worth re-examining it for use once data specific to the intended problem becomes available.

Bibliography

1) Avoiding Numerical Cancellation In The Interior Point Method For Solving Semidefinite Programs, Jos F. Sturm, Mathematical Programming, Volume 95, Number 2 / February, 2003.

2) Stephen J. Wright, Primal-Dual Interior-Point Methods (for linear programming), SIAM, 1997.

3) Interior-Point Polynomial Algorithms in Convex Programming, Nesterov and A. Nemirovskii, SIAM Press, 1994.

4) Anne Greenbaum, Iterative Methods for Solving Linear Systems, SIAM Press, 1997.

5) Weichung Wang and Dianne P. O'Leary, "Adaptive Use of Iterative Methods in Predictor-Corrector Interior Point Methods for Linear Programming", *Numerical Algorithms* 25 (2000) 387-406.

6) Course Notes by Dianne P. O'Leary for AMSC 607 Advanced Numerical Optimization, Fall 2006, online at: http://www.cs.umd.edu/users/oleary/a607/index.html.

7) R. Tyrrell Rockafellar, "Convex Analysis", Princeton University Press, 1997.

8) Bazaraa, Sherali, and Shetty, "Nonlinear Programming: Theory and Algorithms", Wiley, 2006.

9) Nash and Sofer, "Linear and Nonlinear Programming", McGraw-Hill, 1996.

10) Yonathan Bard, "Nonlinear Parameter Estimation", Academic Press, 1974.

11) Borwein and Lewis, "Convex Analysis and Nonlinear Optimization: Theory and Examples", Springer, 2000.

12) Andrzej Ruszcynski, "Nonlinear Optimization", Princeton University Press, 2006.

13) Conejo, Castillo, Minguez, and Garcia-Bertrand, "Decomposition Techniques in Mathematical Programming", Springer, 2006.

14) Gelb, ed., "Applied Optimal Estimation", MIT Press, 1975.

Addendum A

(Primal) Problem:

$Min\ c^T x$ subject to $Ax=b$ with $x \geq 0$

Dual Problem

$Max\ b^T y$ subject to $A^T y \leq c$

(Alternately, subject to $A^T y+z=c$ with $z \geq 0$)

The penalty function augmented version of the problem is:

$Min\ B(x; \mu) = c^T x - \mu \sum ln\ xi$

For which the optimality Conditions are:

$c - \mu X\text{-}1e - A^T y = 0$

$Ax - b = 0$

Collecting all these conditions, we have:

$Ax - b = 0$

$A^T y+z=c$

$z \geq 0$

$x \geq 0$

$c - \mu X\text{-}1e - A^T y = 0 \rightarrow Xz = \mu e$

This produces the system of equations:

$Xz - \mu e = 0$

$Ax - b = 0$

$A^T y+z - c = 0$

We can solve this system using Newton's method, where the solution is incremented by:

$$J(x)\Delta x = -gradient(x)$$

where the Jacobian is:

$$\text{The Jacobian is : } \boldsymbol{J} = \begin{bmatrix} \boldsymbol{Z} & \boldsymbol{0} & \boldsymbol{X} \\ \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A^T} & \boldsymbol{I} \end{bmatrix}$$

Therefore, the Newton step is:

$$\begin{bmatrix} \boldsymbol{Z} & \boldsymbol{0} & \boldsymbol{X} \\ \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A^T} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{y} \\ \Delta \boldsymbol{z} \end{bmatrix} = \begin{bmatrix} \mu e - Xz \\ b - Ax \\ c - A^T y - z \end{bmatrix}$$

If we multiply the first equation by X^{-1} we get:

$$\Delta x + (X\text{-}1Z)\,\Delta z = \mu\, X\text{-}1e - Z \rightarrow \Delta x + \pi\, \Delta z = r$$

Similarly, the next two lines produce:

$$A\,\Delta x = 0$$

$$A^T \Delta y + \Delta z = 0$$

Final Report on:
Improving Performance of the Interior Point Method by Preconditioning
-- Application to Distributed Command and Control --

Addendum B

The data obtained from applying the eight algorithms to the five NETLIB LP test problems is shown below.

Problem Name	Constraint Matrix			Algorithm			Number of Iterations	Execution Time (sec)*	Problem Condition	Variability		
	Number of Rows	Number of Columns	Number of Non-zeros	CG	Precond	QR				x	y	z
afiro	27	51	102	0	0	0	68	0.063	2.95E+13	1879.02	31.4907	59.9621
				0	1	0	68	0.031	8.07E+15	1879.03	31.4908	59.9622
				0	0	1	68	0.063	5.44E+06	1879.02	31.4907	59.9621
				0	1	1	68	0.094	1.48E+09	1879.02	31.4908	59.9622
				1	0	0	176	2.015	5.52E+13	582.691	2.86404	7.10002
				1	1	0	353	7.032	6.87E+15	1347.91	3.44679	15.2213
				1	0	1	195	2.906	7.57E+06	643.202	2.86404	7.10002
				1	1	1	247	4.703	1.37E+09	1422.29	2.83281	15.2213
sc50a	50	78	160	0	0	0	68	0.047	1.35E+17	2598.88	9.30E+10	9.30E+10
				0	1	0	68	0.11	2.19E+19	2596.71	8.48E+10	8.48E+10
				0	0	1	68	0.204	9.22E+15	1730.15	1.61E+09	1.61E+09
				0	1	1	69	0.296	4.08E+18	2596.31	3.17E+09	3.17E+09
				1	0	0	141	7.688	4.08E+11	1348.04	0.808765	0.788882
				1	1	0	204	14.781	1.33E+14	938.131	0.913363	0.76503
				1	0	1	138	9.672	638800	1160.59	0.778781	0.788882
				1	1	1	192	17.625	2.07E+08	697.088	0.537586	0.76503
sc50b	50	78	148	0	0	0	55	0.063	3.29E+18	2995.78	2.20E+11	2.20E+11
				0	1	0	55	0.094	1.15E+20	2440.37	2.18E+11	2.18E+11
				0	0	1	57	0.172	7.01E+15	2991.44	4.80E+09	4.80E+09
				0	1	1	56	0.234	9.89E+18	2992.34	1.08E+10	1.08E+10
				1	0	0	175	5.875	3.19E+11	1376.56	0.760351	0.755577
				1	1	0	211	12.968	1.94E+14	1404.46	1.08116	0.774718
				1	0	1	154	7.172	565487	1595.91	1.15143	0.755577
				1	1	1	223	17.391	3.40E+08	1268.57	1.3201	0.774718
adlittle	56	138	424	0	0	0	290	0.906	1.86E+20	309.867	6.86E+10	6.86E+10
				0	1	0	273	1.141	5.69E+29	8.8064	2.61E+62	2.61E+62
				0	0	1	293	2.656	3.18E+17	308.691	4.91E+09	4.91E+09
				0	1	1	290	3.578	3.03E+22	311.051	1.59E+09	1.59E+09
				1	0	0	13	0.031	1.81E+21	474.402	81.0673	3495.17
				1	1	0	12	0.031	2.80E+26	461.8	21.8697	2434.21
				1	0	1	13	0.078	5.66E+10	474.402	81.0673	3495.17
				1	1	1	12	0.094	1.33E+16	461.8	21.8697	2434.21
blend	74	114	522	0	0	0	79	0.391	1.51E+11	84.6874	65.7568	57.5682
				0	1	0	78	0.359	2.55E+17	86.256	76.3801	75.4527
				0	0	1	77	0.641	388027	83.5595	95.9339	86.292
				0	1	1	78	0.906	6.58E+11	84.4043	87.3398	80.3376
				1	0	0	148	25.609	3.41E+11	99.6998	3.85271	4.73879
				1	1	0	185	41.422	5.25E+15	114.087	5.16216	9.28736
				1	0	1	149	38.984	590107	78.7679	3.80265	4.85423
				1	1	1	179	55.828	5.35E+11	106.136	5.46254	8.84833

* Execution Time is only an approximate measure because some modules were not optimized in favor of maintaining a parallel structure to ensure relative timing was computed in a consistent manner.