FAST SOLUTION OF LINEAR AND CONVEX QUADRATIC PROGRAMMING PROBLEMS WITH TOEPLITZ CONSTRAINTS

PLAMEN KOEV

December 15, 1999

ABSTRACT.
We present a method for exploiting the structure of Toeplitz constraints in linear and some convex quadratic programs. This makes the interior point methods that solve those problems faster by an order of magnitude. Our method works regardless of whether A is square.

1. Introduction

We are interested in solving the primal linear programming problem

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \\ & x \geq 0, \end{array}$$

where A is an $n \times m$ Toeplitz matrix, i.e. has the same entries along all diagonals $A_{ij} = a_{i-j}$ where a is a given vector.

We are also interested in solving the convex quadratic problem

$$\begin{array}{ll} \text{minimize} & c^T x + \frac{1}{2} x^T Q x \\ \text{subject to} & Ax = b, \\ & x \geq 0, \end{array}$$

where the matrix A again has Toeplitz structure and Q is a symmetric positive semidefinite matrix. For the purposes of this presentation we will assume that Q is also a diagonal matrix. In order for our algorithm to work we need a much less stringent restriction on Q, for example it suffices that Q be diagonal, sparse or have low displacement rank [6].

At each iteration the interior point methods for linear or convex quadratic problems solve a system of the form $\begin{bmatrix} S & A \\ A^T & 0 \end{bmatrix} x = b$, where the diagonal matrix S and the vector b change at every iteration. Using the traditional structure ignoring algorithms this would take $O((n+m)^3)$ time. If we exploit the fact that A has Toeplitz structure however we can apply the conjugate gradient method in $O((n+m)^2 \log(m+n))$ time instead.

We do not present a practical problem, however we make some progress towards exploiting the internal matrix structure in the problem, which has been identified as an important goal by Boyd [3; 7,p.18] and El Ghaoui [8, p.1059].

Typeset by $\mathcal{AMS}$-TEX

The rest of this paper is organized as follows. Chapter 2 is devoted to the interior point methods for linear and convex quadratic programming problems. In Chapter 3 we describe how $O(n \log n)$ matrix multiplication works for a Toeplitz matrix. This is the most computationally intensive part of each iteration in conjugate gradient method. In Chapter 4 we describe how to use the fast Toeplitz matrix vector multiplication in order to solve the system using conjugate gradients. Chapter 5 contains the Matlab code and Chapter 6 lists topics for future work.

We use Matlab notation throughout the paper.

2. Interior Point Methods for Linear and Quadratic Problems

Following [1] we consider the primal linear programming problem

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \\ & x \geq 0, \end{array}$$

where the matrix A is Toeplitz structure and its dual

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & A^T y + z = c, \\ & z \geq 0, \end{array}$$

By applying logarithmic barrier function we can remove the inequality constraints. The transformation of the primal yields a nonlinear program

$$\begin{array}{ll} \text{minimize} & c^T x - \mu \sum_{j=1}^{n} \ln x_j \\ \text{subject to} & Ax = b. \end{array}$$

If we fix $\mu > 0$ and apply the barrier transformation to the primal and dual linear programs at a point where $x > 0$ and $z > 0$, we obtain conditions for simultaneous optimality of the primal and dual barrier problems:

$$\begin{array}{l} Ax = b \\ A^T y + z = c \\ XZe = \mu e, \end{array}$$

where $X = \text{diag}(x)$ and $Z = \text{diag}(z)$.

Applying the Newton's method to these equalities we get

$$\begin{array}{l} A\Delta x = \rho, \\ A^T \Delta y + \Delta z = \sigma, \\ Z\Delta x + X\Delta z = \phi, \end{array}$$

where $\rho = b - Ax, \quad \sigma = c - A^T y - z, \quad \phi = \mu e - XZe.$

Eliminating Δz from the system we obtain

$$\begin{bmatrix} -ZX^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma - X^{-1}\phi \\ \rho \end{bmatrix}.$$

We are interested in solving this system fast, provided A is Toeplitz.

Let's now consider the convex quadratic problem

$$\begin{array}{ll} \text{minimize} & c^T x + \frac{1}{2} x^T Q x \\ \text{subject to} & Ax = b, \\ & x \geq 0, \end{array}$$

where the matrix A is again Toeplitz and Q is a symmetric positive semidefinite matrix. The dual problem is

$$\begin{array}{ll} \text{maximize} & b^T y - \frac{1}{2} x^T Q x \\ \text{subject to} & A^T y + z - Qx = c, \\ & z \geq 0, \end{array}$$

Once again applying the logarithmic barrier function we can eliminate the inequality constraints. Simultaneous optimality in the barrier transformed problem requires

$$Ax = b, \quad A^T y + z - Qx = c, \quad XZe = \mu e.$$

The step direction is the solution of the system

$$\begin{bmatrix} X & Z & 0 \\ I & -Q & A^T \\ 0 & A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} \phi \\ \sigma_q \\ \rho \end{bmatrix},$$

where ρ, σ_q and ϕ are respectively the primal infeasibility, dual infeasibility, and noncomplementarity at the current point. The previous definitions of ρ and ϕ still apply, but the dual infeasibility now also depends on the primal variables and is defined as

$$\sigma_q = c + Qx - A^T y - z.$$

We can eliminate Δz from the above system to obtain

$$\begin{bmatrix} -(Q + ZX^{-1}) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma_q - X^{-1}\phi \\ \rho \end{bmatrix}.$$

If we assume Q is diagonal, then we are again faced with a matrix of the form

$$\begin{bmatrix} S & A \\ A^T & 0 \end{bmatrix}.$$

where S is diagonal and A is Toeplitz.

3. Fast Matrix-Vector Multiplication for Toeplitz Matrices

Let A be an $n \times m$ Toeplitz matrix

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \dots & a_{-m+1} \\ a_1 & a_0 & a_{-1} & \dots & a_{-m+2} \\ a_2 & a_1 & a_0 & \dots & a_{-m+3} \\ & \ddots & \ddots & \ddots & \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_{n-m} \end{bmatrix}.$$

We consider the completion of A to a square matrix

$$A_1 = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \dots & a_{-m+1} & 0 & 0 & \dots & 0 & 0 \\ a_1 & a_0 & a_{-1} & \dots & a_{-m+2} & a_{-m+1} & 0 & \dots & 0 & 0 \\ a_2 & a_1 & a_0 & \dots & a_{-m+3} & a_{-m+1} & a_{-m+1} & \dots & 0 & 0 \\ & \ddots & \ddots & \ddots & & & & & & \\ a_{n-2} & a_{n-3} & a_{n-4} & \dots & a_{n-m-1} & a_{n-m-2} & a_{n-m-3} & \dots & a_0 & a_1 \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_{n-m} & a_{n-m-1} & a_{n-m-2} & \dots & a_1 & a_0 \end{bmatrix}.$$

If $y = Ax$ and $x(m+1:n) = 0$ then $y(1:n) = (A_1 x)(1:n)$, i.e. the first n entries of $A_1 x$ are equal to Ax.

Now we can embed the Toeplitz matrix

$$A_1 = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \dots & a_{-n+1} \\ a_1 & a_0 & a_{-1} & \dots & a_{-n+2} \\ a_2 & a_1 & a_0 & \dots & a_{-n+3} \\ & \ddots & \ddots & \ddots & \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_0 \end{bmatrix}$$

$(a_{-m} = \dots = a_{-n+1} = 0)$ into a circulant matrix

$$A_2 = \left[\begin{array}{ccccc|cccc} a_0 & a_{-1} & a_{-2} & \dots & a_{-n+1} & a_{n-1} & a_{n-2} & \dots & a_1 \\ a_1 & a_0 & a_{-1} & \dots & a_{-n+2} & a_{-n+1} & a_{n-1} & \dots & a_2 \\ a_2 & a_1 & a_0 & \dots & a_{-n+3} & a_{-n+2} & a_{-n+2} & \dots & a_3 \\ & \ddots & \ddots & \ddots & & & & & \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_0 & a_{-1} & a_{-2} & \dots & a_{-n+1} \\ -- & -- & -- & - & -- & -- & -- & - & -- \\ a_{-n+1} & a_{n-1} & a_{n-2} & \dots & a_1 & a_0 & a_{-1} & \dots & a_{n-1} \\ a_{-n+2} & a_{n+1} & a_{n-3} & \dots & a_2 & a_1 & a_0 & \dots & a_{n-2} \\ & \ddots & \ddots & \ddots & & & & & \\ a_{-1} & a_{-2} & a_{-3} & \dots & a_{-n+1} & a_{-n+2} & a_{-n+3} & \dots & a_0 \end{array}\right]$$

Now if $y = A_2 x$ and $x(m+1:2n-1) = 0$, then $y(1:m) = Ax$.

For example if

$$A_1 = \begin{bmatrix} 5 & 2 & 7 \\ 4 & 5 & 2 \\ 9 & 4 & 5 \end{bmatrix}$$

then

$$A_2 = \begin{bmatrix} 5 & 2 & 7 & 9 & 4 \\ 4 & 5 & 2 & 7 & 9 \\ 9 & 4 & 5 & 2 & 7 \\ 7 & 9 & 4 & 5 & 2 \\ 2 & 7 & 9 & 4 & 5 \end{bmatrix}$$

However, we can multiply a circulant matrix by a vector in $O(n \log n)$ time using the Fast Fourier Transform (FFT). If $v = A(:,1)$ is the first column of A_2 then [4, p.202]

$$A_2 = F^{-1} \mathrm{diag}(Fv) F,$$

where F is the matrix of the Discrete Fourier Transform. From here we get

$$A_2 x = F^{-1} * (F(v) .* F(x)),$$

i.e. three FFT's and a vector multiply.

So here is how the algorithm works:

1. Form the vectors

$$v = [\, a_0 \quad a_1 \quad a_2 \quad \dots \quad a_{m-1} \quad \text{zeros(n-m,1)} \quad a_{-n+1} \quad a_{-n+2} \quad \dots \quad a_{-1} \,]^T$$

and

$$x = \begin{bmatrix} x \\ \text{zeros}(1, 2n-1-m) \end{bmatrix}$$

2. Compute

$$y = A_2 x = F^{-1} * (F(v) .* F(x)),$$

3. Return $y(1:m)$. $(= Ax)$.

The algorithm works for A^T in a very similar manner.

4. The Conjugate Gradient Method

The conjugate gradient method is a well-studied method and works very well for well conditioned symmetric matrices [5]. The main operation at each step of CG is a matrix vector multiply.

In our case the matrix is $\begin{bmatrix} S & A \\ A^T & 0 \end{bmatrix}$, where A is Toeplitz (and so is A^T) and S is diagonal. If $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ then

$$\begin{bmatrix} S & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} Sx_1 + Ax_2 \\ A^T x_1 \end{bmatrix}.$$

S is a diagonal matrix, so forming Sx_1 is just diagonal scaling taking $O(n)$ operations, Ax_2 and $A^T x_1$ are Toeplitz matrix vector multiplies, each of which costs at most $O((n+m)\log(n+m))$ for a total per-iteration cost of $O((n+m)\log(n+m))$.

This is an order of magnitude faster than the traditional CG algorithm which does not exploit the Toeplitz structure of A.

This method will work just as fast if we can compute Sx_1 in not more than $O(n \log n)$ time. Since S is square there is a wider class of matrices for which this is possible e.g. Toeplitz, sparse, Vandermonde and other matrices with low displacement rank.

5. The Matlab Code

```
% This is just test code

A=[5 2 7 ; 4 5 2; 9 4 5; 7 9 4; 2 7 9];
x=[1 2 3 4 5]';
```

```
Ax(A',x)
A'*x
[n,m]=size(A);
s=[1 2 3 4 5]';
B=[diag(s) A; A' zeros(m)];
y=[x;1;2;3];
z1=cg(A,s,y) % the computed solution with the new algorithm
z2=B\y % for comparison the true result with the old algorithm
```

```
% Computes the product Ax for a Toeplitz matrix A and a vector x
% using the FFT in O((n+m)log(n+m)) time
%
% function y=Ax(A,x);

function y=Ax(A,x);
[n,m]=size(A);
v=[A(:,1); zeros(abs(n-m),1); A(1,m:-1:2)'];
x=[x; zeros(2*max(n,m)-1-m,1)];
y=ifft(fft(v).*fft(x));
y=real(y(1:n));
```

```
% Solves
% [S  A]x=b
% [A' 0],
% where S=diag(s)
% using conjugate gradients
%
% function x=cg(A,s,b);

function x=cg(A,s,b);
[n,m]=size(A);
k=0; x0=0; r0=b; p1=b; z=b;
while (norm(r0)>1e-6)
   k=k+1;
   z(n+1:m+n)=Ax(A',p1(1:n));
   z(1:n)=s.*p1(1:n)+Ax(A,p1(n+1:m+n));
   v1=(r0'*r0)/(p1'*z);
   x1=x0+v1*p1;
   r1=r0-v1*z;
   p2=r1+(r1'*r1)/(r0'*r0)*p1;
   x0=x1;
   r0=r1;
   p1=p2;
end
x=x1;
```

5. Conclusions and Future Work

We have described how to implement the iterative step for interior point methods for linear and some convex quadratic programming problems with Toeplitz constraints.

Future could be devoted to the following topics:

1. Testing the method for stability and possibly implementing the minimal residual algorithm instead (MINRES) in the cases when the system is ill-conditioned. Our system is not s.p.d. and the conjugate gradient method may be unstable if the matrix is ill-conditioned.

2. Extending these algorithms to other displacement structures, e.g. Vandermonde, Cauchy, etc.

3. Extending these algorithms to work with semidefinite programs where the matrices have displacement structure, e.g. are all Toeplitz.

References

1. Robert Vanderbei, Tamra Carpenter, Symmetric Indefinite Systems for Interior Point Methods, Mathematical Programming 58 (1993) 1-32.
2. J. Chun, T. Kailath, Generalized displacement structure for Block-Toeplitz, Toeplitz-Blockand Toeplitz-derived matrices. NATO ASI series, vol.F 70, numerical linear algebra, digital signal processing and parallel algorithms, 1991
3. L. Vandenberghe, S. Boyd, Semidefinite Programming, SIAM Review 38 (1996) 49-95
4. G. Golub, C. Van Loan, Matrix Computations, Third Edition, John Hopkins University Press, 1996

5. J. Demmel, Applied Numerical Linear Algebra, SIAM, 1996
6. T. Kailath, A. Sayed, Displacement Structure: Theory and Applications, SIAM Review 37 (1995), 297-386
7. L. Vandenberghe, S. Boyd, SP, Software for Semidefinite Programming, User's Guide
8. L. El Ghaoui, H. Lebret, Robust Solutions to Least Squares Problems with Uncertain Data, SIAM J. Matrix Anal. Appl. 18 (1997), 1035-1064.