NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2007; **14**:695–712
Published online 4 September 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla.547

An efficient linear programming solver for optimal filter synthesis

Jihong Ren[1], Chen Greif[2,*,†] and Mark R. Greenstreet[2]

[1]*Rambus, Los Altos, CA 94022, U.S.A.*
[2]*Department of Computer Science, University of British Columbia, Vancouver, BC, Canada V6T 1Z4*

SUMMARY

We consider the problem of l_∞ optimal deconvolution arising in high data-rate communication between integrated circuits. The optimal deconvolver can be found by solving a linear program for which we use Mehrotra's interior-point approach. The critical step is solving the linear system for the normal equations in each iteration. We show that this linear system has a special block structure that can be exploited to obtain a fast solution technique whose overall computational cost depends mostly on the number of design variables, and only linearly on the number of constraints. Numerical experiments validate our findings and illustrate the merits of our approach. Copyright © 2007 John Wiley & Sons, Ltd.

Received 25 October 2006; Revised 22 June 2007; Accepted 23 June 2007

KEY WORDS: high-speed digital interconnect; filter design; l_∞ optimization; interior-point methods; normal equations; direct linear solvers; Schur complement; Sherman–Morrison–Woodbury formula

1. INTRODUCTION

Integrated circuits ('chips') have improved dramatically in both speed and density for several decades. The printed circuit board technology that provides connections between different chips, on the other hand, has achieved only modest improvements. To provide adequate inter-chip communication performance, designers rely on a variety of circuit design techniques to compensate for the limitations of the off-chip interconnect. One of the most important of these techniques is equalization.

*Correspondence to: Chen Greif, Department of Computer Science, University of British Columbia, Vancouver, BC, Canada V6T 1Z4.
†E-mail: greif@cs.ubc.ca

Contract/grant sponsor: BC-ASI
Contract/grant sponsor: Intel
Contract/grant sponsor: NSERC
Contract/grant sponsor: SUN Microsystems

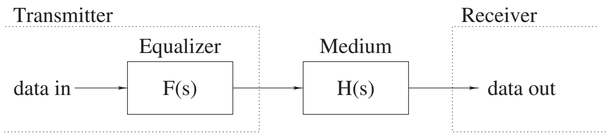

Figure 1. Block diagram of an equalized transmission channel (from [1, p. 364]).

The basic idea behind equalization is simple: the process by which off-chip interconnect degrades digital signal integrity is essentially a linear convolution; an equalization filter is designed to have a response that approximates the inverse of the response of the interconnect. Figure 1 shows a typical application of equalization for high-speed, digital interconnect.

We refer to the collection of wires, connectors, etc. that convey signals from the transmitter to the receiver as the *medium*. Often this collection will have more than one wire to provide enough total bandwidth. We refer to each wire as a *line*. These lines are coupled through electric and magnetic fields. This coupling creates *crosstalk* where signals sent on one line interfere with signals sent on other lines. The velocity at which signals propagate along the lines depends on their frequencies. This creates *dispersion* where data sent at one time interferes with data sent at other times. Furthermore, attenuation over a line depends strongly on frequency, and reflections occur due to imperfect impedance matching at connectors, vias, and other discontinuities in the line. These processes are linear—they can be expressed as two-dimensional convolutions: one dimension is space, interference between different wires; and the other is time, interference between values sent at different times. An equalization filter performs a deconvolution to enable transmission of data at greater rates than would be possible on the bus by itself.

In current practice, the highest bit rates per wire are achieved by serial links using various forms of equalization filters (e.g. [2]). These serial links require a significant amount of power and chip area. Thus, designs requiring very large bandwidths such as high-performance microprocessors use buses with 64–256 or more lines. Equalization has been used for buses (e.g. [3]), but in general equalization for buses has been less aggressive than that for serial links. Our current work enables the synthesis of *optimal* equalization filter for all of these designs. In addition to enabling higher performance designs, our methods enable designers to determine the limits of various approaches and to better understand the trade-offs involved.

We refer to the combination of the medium and the filters as a *channel*. If we could implement a filter whose response is the exact inverse of that of the medium, then data could be conveyed with perfect fidelity at arbitrarily high rates. However, limitations of power and switching rates of the filter force the use of approximate filters. The equalizer synthesis problem is to determine the optimal equalization filter under constraints of filter size, speed, and power consumption.

To solve the problem, one can seek to minimize the norm of the difference between the received signal and a delayed version of the transmitted signal. Link designers have often used the l_2 norm for this because the problem is straightforward to formulate and solve. Moreover, adaptive schemes such as sign–sign least mean squares (SS-LMS) are efficient and require minimal hardware support. For these reasons, l_2-based approaches are commonly used to design equalizers for telephone subscriber systems [4], wireless communication [5, Chapter 8] and more recently for high-speed serial links [6, 7].

High-speed digital communication requires extremely low bit error rates (typically less than 10^{-15}). In this case, the worst-case error is more relevant than average-case error (see Section 2).

Thus, the commonly used specifications for digital signal integrity such as eye masks and eye diagrams are worst-case measures. This leads to an l_∞-based optimization approach that directly optimizes eye masks, and achieves better signal integrity than l_2 methods. In [8], the advantages of the l_∞ approach over l_2 are demonstrated for a wide range of filter configurations.

In this paper, we show that our l_∞ formulation for equalization filter synthesis can be solved by exploiting the special structure of the linear systems that arise during the iterations of an interior-point linear programming solver. The associated matrices contain a large block that can be expressed as the sum of a diagonal matrix and a low rank block diagonal matrix. This mathematical structure allows for fast direct inversion. We express the matrix of the normal equations as a 2×2 block matrix, and use the Sherman–Morrison–Woodbury formula to obtain a factorization-free formula for the inverse of the (1,1) block. This is followed by forming the Schur complement in an efficient manner, taking into account the block structure of the matrices involved. The number of constraints in a linear programming formulation of l_∞ optimal deconvolution is large; our algorithm allows for their elimination with minimal computational overhead. Hence, the overall cost of the linear system solver in most configurations is cubic in the number of design variables (which is relatively small), and is merely linearly dependent on the number of constraints (which can be quite large). For example, for a typical 32-bit wide bus with an equalizer filter length of 4, the number of design variables is only 128 while the number of constraints could be tens of thousands depending on other design parameters.

Methods that exploit the sparsity structure of the matrix in the context of interior-point methods have been proposed for a variety of applications [9–11] and have shown to be very effective. The approach we are proposing shows that substantial savings can be made for an application in the field of high-speed digital interconnect, by taking full advantage of the special structure of the underlying matrices. Our solution procedure is fast, memory efficient, and stable. Because the structure that we exploit is rooted in the l_∞ formulation, our approach is likely to be applicable to l_∞ deconvolution problems arising in other applications as well.

The remainder of this paper is structured as follows. In Section 2, we describe the equalization filter synthesis problem and formulate it as a linear program. In Section 3, we present our linear system solver, discuss the computational work involved, and show that the decomposition that the solver is based upon does not significantly affect the conditioning of the problem. We give a short example in Section 4 and draw some conclusions in Section 5.

2. EQUALIZATION AND THE LINEAR PROGRAM

For digital communication, ‘eye diagrams’ (so named because of their shape) are commonly used to capture signal integrity requirements. As shown in Figure 2, an eye diagram is obtained by overlaying the received signal over multiple symbol periods for a large number of different input patterns. Ideally, all possible input patterns would be considered. The eye is defined by the highest and lowest levels received at each time point over the bit period. The intuition is that if a channel conveys discrete values, then the transmitted data will be received without error if each received value is close enough to its target value. From the linearity of the channel, it is sufficient to consider only the response to the highest and lowest possible values on each line of the medium. The gap between the highest allowed low signal and the lowest allowed high signal is referred to as the ‘eye height’. Likewise, the time interval in which high and low values are distinct is referred to as the eye width. Simplistically, increasing eye height or eye width corresponds to improving

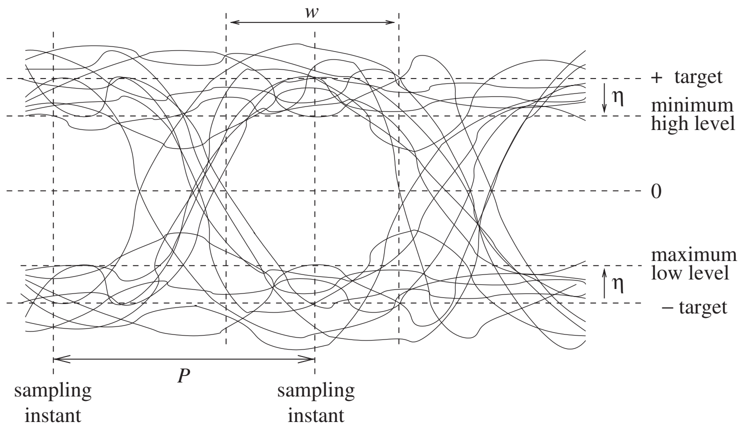

Figure 2. An eye diagram.

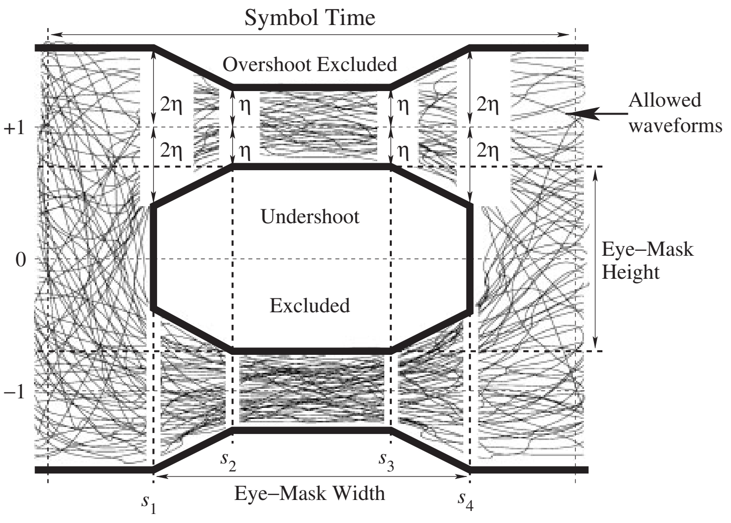

Figure 3. A parameterized eye mask.

overall signal integrity. We can also include constraints to set the maximum value of a logically high signal or the minimum value of a logically low signal. Such constraints allow our approach to be generalized to interconnects that use multi-level signalling, i.e. three or more signal levels per symbol transmitted. Overshoot constraints can also be used to prevent operating conditions that would damage or degrade the hardware. In all these cases, the signal integrity is determined by the extremal signals. Thus, eye masks are an l_∞ measure of signal integrity.

Eye masks can be parameterized, for example, by making the height required at each sample time a linear function of a parameter, η as shown in Figure 3. In this example, minimizing η optimizes signal integrity. In this paper, we assume that overshoot and undershoot constraints are symmetrical about zero. This simplifies the presentation; however, the more general formulation with asymmetric constraints can be solved with the same efficiency and robustness as the symmetric case.

As noted above, the response of inter-chip interconnect is linear in the inputs; it can be expressed as a convolution of the inputs applied to the interconnect with the impulse response of the medium. Thus, equalization filter synthesis is a deconvolution problem constrained by the size, speed, and power limitations of the filter. The objective function is based on extremal responses; thus, this is a constrained, l_∞, deconvolution problem.

Equalization filters can be used at the input of the medium, the output of the medium, or in combination. For simplicity, we only consider equalizers that are at the input of the medium in this paper. Such equalizers are called pre-equalizers. Filter synthesis for more general configurations are considered in [12, 13]. For high-speed digital interconnect, these filters must operate at very high data rates; thus, they must be simple. In practice, this means that equalization filters are FIR (finite impulse response [14]): the output of the filter is a weighted sum of the inputs.

We make two more observations about typical, high-speed links before we formulate the optimization problem. First, we assume that the medium is linear and time invariant. Thus, the response to an arbitrary input on a particular line can be derived from the response to a pulse on that line. Second, we recognize that the response of the medium can be distinct for each line due to variations in design, manufacturing, geometrical constraints, etc. Thus, we do not assume a response that is invariant under spatial shifts. We revisit these assumptions of time-shift invariance and space-shift non-invariance in Section 5.

Suppose an l_2 formulation is sought. In the absence of additive noise, if all symbols are equally likely, then minimizing the mean square error of the received signal is the same as minimizing the dispersion and crosstalk components of the channel's impulse response. This is called zero-forcing [5, Chapter 9]:

$$f_{\mathrm{ZF}} = \arg\min_{f}\{\|Hf - e_\delta\|_2\} \tag{1}$$

where f ranges over all possible vectors of filter coefficients, δ is the delay of the channel and e_δ denotes column δ of the identity matrix.

Alternatively, we can achieve l_∞ optimality and formulate optimal filter synthesis as a linear programming problem [12]. The output of the medium is the convolution of the impulse response of the medium with the input signal. We can write this as a

$$\mathsf{out} = H\,\mathsf{in} \tag{2}$$

where in is a vector of input values with an element for each line of the medium and each input time; out is a vector of medium output values, and H is the pulse response of the medium. Ideally, the response of the channel would be a delayed version of the input, $\mathsf{out}_{\mathsf{ideal}} = D\,\mathsf{in}$, where D is a delay matrix (a matrix with ones along the off-diagonal corresponding to the desired delay, and zeros everywhere else). Let F be the matrix for convolving the input to the channel with the pre-equalizing filter. The l_∞ optimal filter synthesis problem is

$$F_{\mathrm{opt}} = \arg\min_{F}\max_{\mathsf{in}} \|D\mathsf{in} - HF\mathsf{in}\|_\infty \tag{3}$$

We note that the response is linear in the filter coefficients, and Equation (3) can be rewritten [13, Chapter 3.4.1] as

$$f_{\mathrm{opt}} = \arg\min_{f}\max_{\mathsf{IN}} \|D\mathsf{in} - H\mathsf{IN}f\|_\infty \tag{4}$$

where f is a vector whose elements are the coefficients of the filter, and IN is a matrix for convolving the filter with the input values, in. In general, solving linear programs is computationally challenging and an efficient solution method should exploit the specific properties of the matrices involved.

The key observation for formulating our deconvolution problem as a linear program is that the worst-case error occurs when each input takes on an extremal value from its domain. Because the system is linear, we can assume that these extremal values are $+1$ and -1. Furthermore, we can compute the response of the filter and medium to individual inputs of $+1$ on each line. The worst-case response for any line and output time is then obtained by computing the sum of the absolute values of the errors when $+1$ inputs are applied. Thus, our formulation for deconvolution consists of three parts:

1. Compute the responses to $+1$ inputs as a function of the filter coefficients.
2. Compute the absolute values of the error terms.
3. Compute the sum of the absolute values of the error terms for each output line and each output time.

These steps are described in more detail in [12, 13].

The computations listed above can be expressed conveniently as a linear program for l_∞ optimal deconvolution. Let w_{bus} be the number of lines in the bus, n_{bus} be the length of the impulse response of the medium (in bit times), and n_{fir} be the length of the impulse response of the filter (in bit times). Let k_{fir} be the number of filter coefficients.

As noted above, an eye mask may have several sample points for each bit period. Let k_{mask} be the number of sample points in the eye mask. To determine the response at each sample point, we convolve the input, filter impulse response, and medium impulse response for each sample point. Thus, an input pulse with a width of one bit time on one line contributes to the outputs each of the w_{bus} lines at $(n_{\text{bus}}+n_{\text{fir}}+1)k_{\text{mask}}-1$ different *sample times*. For convenience, we round it up to be $(n_{\text{bus}}+n_{\text{fir}}+1)k_{\text{mask}}$. We construct G, a $w_{\text{bus}}k_{\text{mask}}(n_{\text{bus}}+n_{\text{fir}}+1)\times k_{\text{fir}}$ matrix such that Gf gives the output on each line at each sample time of the response to a separate, single-bit pulse on each line. Matrix G is related to the H matrix from Equation (4) by

$$G=\begin{bmatrix} H\mathsf{IN}_1 \\ H\mathsf{IN}_2 \\ \vdots \\ H\mathsf{IN}_{w_{\text{bus}}} \end{bmatrix} \tag{5}$$

where IN_1 is an input with a pulse at time 0 on line 1 and zeros for all other lines and all other times, and likewise for the other IN_j's. Due to the linearity of the bus, these responses provide a basis from which we can determine the response to any input pattern. In particular, we can determine the worst-case input pattern and the responses that it generates.

We partition the rows of G to form two matrices, G_u and G_d. Matrix G_u computes the 'undisturbed' response of the channel, the output of the medium at the desired delay in response to a single-bit pulse on the corresponding line of the input. Likewise, G_d computes the disturbances, the output of the medium on all other lines and for the same line at all other times. With this partitioning, our linear program for l_∞ optimal filter synthesis is

$$\begin{aligned} &\min_{f,d,\eta} \quad \eta \\ &\text{s.t.} \quad \begin{bmatrix} -I & G_d & 0 \\ -I & -G_d & 0 \\ W & G_u & -a \\ W & -G_u & -a \end{bmatrix} \begin{bmatrix} d \\ f \\ \eta \end{bmatrix} \leqslant \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} \end{aligned} \tag{6}$$

In the first two rows of constraints $G_d f$ computes the disturbances, the coupling between lines and responses at times other than the desired delay. Requiring d to be greater than both these responses and their negation sets the components of d (at optimality) to the absolute value of these disturbance terms. The next two rows compute the overshoot and undershoot, respectively. In particular, $G_u f$ computes the desired (i.e. *undisturbed*) responses of the channel, and matrix W computes the sum of the disturbance terms that affect the output to determine the worst-case disturbance. We describe the detailed structure of W later in this section (see Equation (9)). Note that the last row, $Wd - G_u f \leqslant -1 + a\eta$, computes the worst-case undershoot by determining the highest possible value at each sample time for a logical low signal. By the linearity of the filter and channel and our assumption that logical low and high values are -1 and $+1$, respectively, this is equivalent to computing the lowest possible value for a logically high signal. The vector a sets the relative threshold values at each sample point as depicted in Figure 3.

In practice, we also include constraints to limit the maximum output of the filter. These constraints have the same form as those shown here, but, in practice, there are far fewer of them. Thus, we focus on the simple form for the linear program from Equation (6). The approach that we present here applies to the more general version [12], and these generalizations do not affect the overall complexity or stability of our algorithm.

The sizes of the vectors and matrices in this problem depend on w_{bus}, n_{bus}, n_{fir}, k_{fir} and k_{mask} as defined above. Matrix G_u computes one output for each line, the output at the desired bit time in response to a pulse on the corresponding input. Thus, G_u has $w_{\text{bus}}k_{\text{mask}}$ rows. Likewise, G_u has k_{fir} columns, one for each coefficient of the filter. Matrix G_d also has k_{fir} columns, and one row for each disturbance term. Each line may have distinct electrical properties; any line can couple to any other line; and the coupling coefficients are not necessarily symmetric. Thus, G_d computes distinct coupling terms for each pair of lines. With these assumptions, the number of disturbances is

$$k_{\text{disturb}} = w_{\text{bus}}^2 k_{\text{mask}} n_{\text{channel}} \tag{7}$$

where

$$n_{\text{channel}} = (n_{\text{bus}} + n_{\text{fir}})k_{\text{mask}} - 1 \approx (n_{\text{bus}} + n_{\text{fir}})k_{\text{mask}} \tag{8}$$

is the length of the impulse response of the channel in eye-mask *sample times*. For simplicity, we use the approximation $n_{\text{channel}} \approx (n_{\text{bus}} + n_{\text{fir}})k_{\text{mask}}$ in the remainder of this presentation.

To describe the structure of G_u and G_d, we return to matrix G. Row $i w_{\text{bus}} n_{\text{channel}} k_{\text{mask}} + j w_{\text{bus}} + q$ of G computes the output of the medium on line i at *sample-time* j in response to an input to the filter on line q at *bit-time* 0. Note that a filter coefficient maps a filter input at one time to a filter output (i.e. input of the medium) at another time. Thus, on average only a fraction of $1/w_{\text{bus}}$ of the filter coefficients compute responses from inputs on any particular line (e.g. line q). It follows then that roughly a fraction of $1/w_{\text{bus}}$ of the coefficients of each row are non-zero. This reasoning applies to all rows of G; therefore, matrices G_u and G_d are relatively sparse: a fraction of $\sim 1/w_{\text{bus}}$ of their elements are non-zero.

Matrix W computes the sum of the disturbances. It is $w_{\text{bus}}k_{\text{mask}} \times k_{\text{disturb}}$ and has the following structure:

$$W = \begin{bmatrix} 1\cdots1 & & & & \mathbf{0} \\ & 1\cdots1 & & & \\ & & 1\cdots1 & & \\ & & & \ddots & \\ \mathbf{0} & & & & 1\cdots1 \end{bmatrix} \tag{9}$$

More formally,

$$w_{i,j} = \begin{cases} 1 & \text{if } (i-1)k_{\text{block}} < j \leqslant i k_{\text{block}} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $k_{\text{block}} = k_{\text{disturb}}/(w_{\text{bus}}k_{\text{mask}})$ is the length of each non-overlapping stripe of 1's in W. As shown in the next section, it is this structure that allows for an efficient linear system solver for this l_∞ optimization problem.

The complete linear program has $k_{\text{fir}}+k_{\text{disturb}}+1$ variables, and $2(w_{\text{bus}}k_{\text{mask}}+k_{\text{disturb}})$ constraints. As a typical example, consider synthesizing an equalizer for a bus with 32 wires ($w_{\text{bus}}=32$), a bus impulse response that is 10 bits long ($n_{\text{bus}}=10$). Assume that the filter for each wire considers the value on that wire and its two nearest neighbors in each direction, and that the filter has an impulse response that is three bit periods long ($n_{\text{fir}}=3$). Assume that there are eight sample points in the eye mask ($k_{\text{mask}}=8$). For this example, the number for filter coefficients, k_{fir}, (i.e. the number of design variables) is 462, and the number of disturbance terms, k_{disturb} is 106 496. The linear program has 106 959 variables and 213 504 constraints. The size of the linear program is dominated by the calculations of the disturbance terms; the actual number of filter coefficients is relatively insignificant. This property of the linear programming approach makes a direct solution of the linear program much less efficient than approximating the solution with least-squares optimization. Fortunately, the structure of the constraint matrix can be exploited to obtain a linear program solver that is competitive with least-squares techniques while retaining the advantages of solving the actual, l_∞ optimization problem. The next section presents this approach in detail. For notational convenience, in the sequel the sizes of the matrices involved will be defined by the five parameters:

$n \equiv k_{\text{disturb}}$: the number of 'disturbances' to be mitigated by the deconvolution.
$m \equiv w_{\text{bus}}k_{\text{mask}}$: the number of points at which the deconvolution quality is specified.
$k \equiv k_{\text{fir}}$: the number of FIR coefficients in the deconvolver.
$\ell \equiv k_{\text{block}}$: the length of the strips of W; in other words, the number of disturbances accumulated per specification point. Note that $\ell = n/m$.
$\rho \equiv 1/w_{\text{bus}}$: the relative density of G_u and G_d, i.e. the ratio of their non-zero elements to the total number of elements.

For the example above, $n = 106\,496$, $m = 256$, $\ell = 416$, and $\rho = \frac{1}{32}$.

3. AN EFFICIENT LINEAR SYSTEM SOLVER

Let us denote the constraint matrix in Equation (6) by A:

$$A = \begin{bmatrix} -I & G_d & 0 \\ -I & -G_d & 0 \\ W & G_u & -a \\ W & -G_u & -a \end{bmatrix} \tag{11}$$

Here, the identity matrix I is $n \times n$, W is $m \times n$, and G_u and G_d are $m \times k$ and $n \times k$, respectively. The two zero vectors are of length n, and a is of length m. Matrix A is thus of size $(2n + 2m) \times (n + k + 1)$.

For solving the problem, we will use an interior-point method. A nonlinear function is defined, whose roots coincide with the first-order optimality conditions (also known as KKT conditions). Newton steps are then taken to find the roots of this function, where necessary non-negativity constraints are imposed. A detailed description of the theory of interior-point methods for linear and quadratic programming can be found, for example, in [15–17].

In practice, applying the Newton method in its basic form does not always yield convergence. A possible way to overcome this is by performing predictor and corrector solves at each step to ensure sufficient progress toward the optimal point. After a Newton step is computed, another system with a different right-hand side is set up. The adjustment corresponds to correcting the iterates so that they progress along what is known as the *central path*. Let $x = [d, f, \eta]$ (in transposed form). Then, in each Newton step of Mehrotra'a algorithm, we solve a linear system corresponding to the normal equations:

$$A^{\mathrm{T}} \Lambda A x = y \tag{12}$$

where Λ is a positive diagonal matrix whose elements are updated in each LP iteration. Note that this formulation is based on forming the normal equations for the *dual* problem; see (6).

This methodology was introduced in [18] and has gained much popularity among practitioners, mainly due to the rapid convergence that has been observed in various implementations of the algorithm [19, 20]. As is noted in [19], Mehrotra's algorithm almost always improves the performance of the primal–dual algorithm, possibly in a dramatic manner when the problem size is large.

The number of variables in the normal equations grows quadratically with the width of the medium. For large problems, a general-purpose solver that does not take advantage of the structure of the matrix may be computationally costly and impractical. We present below a direct solution method that exploits the special structure of the linear system and proves to be extremely efficient.

Based on the sizes of the matrices, we split the diagonal matrix Λ as follows:

$$\Lambda = \begin{bmatrix} \Lambda_1 & 0 & 0 & 0 \\ 0 & \Lambda_2 & 0 & 0 \\ 0 & 0 & \Lambda_3 & 0 \\ 0 & 0 & 0 & \Lambda_4 \end{bmatrix} \tag{13}$$

where Λ_1 and Λ_2 are $n \times n$ and Λ_3 and Λ_4 are $m \times m$.

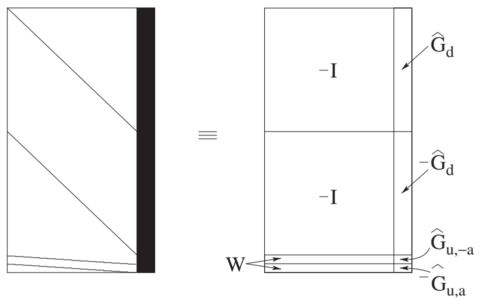

Figure 4. Sparsity pattern of the constraint matrix A.

Given that the last column of A is really just a vector, and that some of its blocks are similar in structure to one another, it is convenient to represent the matrix in a more compact form. To this end, define

$$\widehat{G}_d = [G_d \;\; 0], \quad \widehat{G}_{u,-a} = [G_u \;\; -a], \quad \widehat{G}_{u,a} = [G_u \;\; +a] \tag{14}$$

Matrix A can now be written as

$$A = \begin{bmatrix} -I & \widehat{G}_d \\ -I & -\widehat{G}_d \\ W & \widehat{G}_{u,-a} \\ W & -\widehat{G}_{u,a} \end{bmatrix} \tag{15}$$

Figure 4 illustrates of the sparsity structure of this matrix.

The matrix of the normal equations can now be written as

$$A^{\mathrm{T}}\Lambda A = \begin{bmatrix} M_1 & M_2 \\ M_2^{\mathrm{T}} & M_3 \end{bmatrix} \tag{16}$$

where

$$\begin{aligned} M_1 &= \Lambda_1 + \Lambda_2 + W^{\mathrm{T}}(\Lambda_3 + \Lambda_4)W \\ M_2 &= (\Lambda_2 - \Lambda_1)\widehat{G}_d + W^{\mathrm{T}}(\Lambda_3\widehat{G}_{u,-a} - \Lambda_4\widehat{G}_{u,a}) \\ M_3 &= \widehat{G}_d^{\mathrm{T}}(\Lambda_1 + \Lambda_2)\widehat{G}_d + \widehat{G}_{u,-a}^{\mathrm{T}}\Lambda_3\widehat{G}_{u,-a} + \widehat{G}_{u,a}^{\mathrm{T}}\Lambda_4\widehat{G}_{u,a} \end{aligned} \tag{17}$$

By construction, M_1 is $n \times n$, M_2 is $n \times (k+1)$, and M_3 is $(k+1) \times (k+1)$. Figure 5 illustrates the sparsity pattern of $A^{\mathrm{T}}\Lambda A$.

3.1. Factorization-free construction of M_1^{-1}

The block structure of $A^{\mathrm{T}}\Lambda A$ lends itself naturally to a block elimination procedure. In particular, since M_1 is block diagonal, it makes sense to solve the system by forming the Schur complement

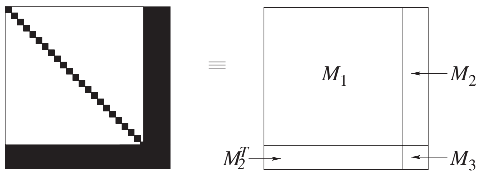

Figure 5. Sparsity pattern of $A^{\mathrm{T}}\Lambda A$.

and then solving for the remaining unknowns:

$$\begin{aligned} Sx_2 &= y_2 - M_2^{\mathrm{T}} M_1^{-1} y_1 \\ M_1 x_1 &= y_1 - M_2 x_2 \end{aligned} \tag{18}$$

where

$$S = M_3 - M_2^{\mathrm{T}} M_1^{-1} M_2 \tag{19}$$

and $x = [x_1, x_2]$, $y = [y_1, y_2]$, with x_1, y_1 vectors of length n and x_2, y_2 vectors of length $k+1$.

The Schur complement S is $(k+1)\times(k+1)$, i.e. much smaller than the matrix of the normal equations. Recalling that in the application we are discussing $k \equiv k_{\mathrm{fir}}$ is the number of filter coefficients, the size of the Schur complement corresponds to the size of the original design problem. The computational cost is determined primarily by the effort required to invert M_1, form S, and solve for S. Fortunately, the special structure of W allows us to derive expressions for the entries of M_1^{-1} which can be computed directly (i.e. without factoring the matrix) and efficiently. We have the following two useful results.

Proposition 3.1
Given a diagonal matrix D of size $m \times m$ and matrix W defined in Equation (9), the $n \times n$ matrix $W^{\mathrm{T}} D W$ is block diagonal with m blocks, each of size $\ell \times \ell$. The ith block of this matrix is a rank-1 matrix all of whose entries are equal to d_i, the ith diagonal entry of D.

Proposition 3.2
Given a diagonal matrix E of size $n \times n$ with diagonal entries e_i, the $m \times m$ matrix $C = WEW^{\mathrm{T}}$ is diagonal, and its diagonal entries, c_i, are

$$c_i = \sum_{j=1}^{\ell} e_{(i-1)\ell + j}$$

The proofs of Propositions 3.1 and 3.2 are straightforward and rely on the special structure of W, namely that it is a matrix of horizontal, non-overlapping stripes, each of size ℓ, and we have m such stripes. Indeed, the structure of W is to an extent reminiscent of that of Galerkin-type prolongation and restriction operators applied in multigrid-type applications [21].

Let

$$\begin{aligned} \Lambda_{12} &= \Lambda_1 + \Lambda_2 \\ \Lambda_{34} &= \Lambda_3 + \Lambda_4 \end{aligned} \tag{20}$$

and define

$$V = (\Lambda_{34})^{1/2} W$$

In the sequel, when necessary we will denote the ith diagonal entry of Λ_{12} by $(\Lambda_{12})_i$, and similarly for Λ_{34}. We can write

$$M_1 = \Lambda_{12} + V^{\mathrm{T}} V$$

and by Proposition 3.1 we have that $V^{\mathrm{T}}V \equiv W^{\mathrm{T}}\Lambda_{34}W$ is block diagonal with m blocks of size ℓ, and all the elements in the ith sub-block have the value $(\Lambda_{34})_i$. Thus, each block of M_1 is a rank-1 update of a diagonal matrix. This allows us to apply the Sherman–Morrison–Woodbury formula [22] to obtain a closed-form formula for M_1^{-1} that involves only diagonal matrix inversion and low-rank updates. Define

$$B = I + V\Lambda_{12}^{-1}V^{\mathrm{T}} \tag{21}$$

Then, we have

$$\begin{aligned} M_1^{-1} &= \Lambda_{12}^{-1} - \Lambda_{12}^{-1}V^{\mathrm{T}}(I + V\Lambda_{12}^{-1}V^{\mathrm{T}})^{-1}V\Lambda_{12}^{-1} \\ &= \Lambda_{12}^{-1} - \Lambda_{12}^{-1}V^{\mathrm{T}}B^{-1}V\Lambda_{12}^{-1} \end{aligned} \tag{22}$$

Proposition 3.3
The $m \times m$ matrix B is diagonal and its diagonal entries are given by

$$b_i = 1 + (\Lambda_{34})_i \sum_{i=1}^{\ell} (\Lambda_{12})^{-1}_{(i-1)\ell+j}$$

Proof
Write $V\Lambda_{12}^{-1}V^{\mathrm{T}}$ as $\Lambda_{34}^{1/2}W\Lambda_{12}^{-1}W^{\mathrm{T}}\Lambda_{34}^{1/2}$ and apply Proposition 3.2 to $W\Lambda_{12}^{-1}W^{\mathrm{T}}$. □

3.2. Construction of the Schur complement and computational work

Using the formula for M_1^{-1} from (22) we obtain

$$\begin{aligned} S &= M_3 - M_2^{\mathrm{T}}M_1^{-1}M_2 \\ &= M_3 - [M_2^{\mathrm{T}}\Lambda_{12}^{-1}M_2 - (V\Lambda_{12}^{-1}M_2)^{\mathrm{T}}B^{-1}(V\Lambda_{12}^{-1}M_2)] \end{aligned} \tag{23}$$

We expand M_2 and M_3 according to their definitions in Equation (17) to obtain

$$M_3 - M_2^{\mathrm{T}}\Lambda_{12}^{-1}M_2 = S_1 - (S_2 + S_2^{\mathrm{T}}) \tag{24}$$

where

$$\begin{aligned} S_1 &= \widehat{G}_d^{\mathrm{T}}(\Lambda_{12} - (\Lambda_2 - \Lambda_1)^2\Lambda_{12}^{-1})\widehat{G}_d \\ &\quad + \widehat{G}_{u,-a}^{\mathrm{T}}\Lambda_3\widehat{G}_{u,-a} + \widehat{G}_{u,a}^{\mathrm{T}}\Lambda_4\widehat{G}_{u,a} \\ &\quad + (\widehat{G}_{u,-a}^{\mathrm{T}}\Lambda_3 - \widehat{G}_{u,a}^{\mathrm{T}}\Lambda_4)W\Lambda_{12}^{-1}W^{\mathrm{T}}(\Lambda_3\widehat{G}_{u,-a} - \Lambda_4\widehat{G}_{u,a}) \\ S_2 &= -(\widehat{G}_{u,-a}^{\mathrm{T}}\Lambda_3 - \widehat{G}_{u,a}^{\mathrm{T}}\Lambda_4)W\Lambda_{12}^{-1}(\Lambda_2 - \Lambda_1)\widehat{G}_d \end{aligned} \tag{25}$$

Forming S_1 and S_2 is computationally advantageous, as it saves redundant floating point operations as well as storage. Indeed, we wish to avoid forming M_2 explicitly because it is dense. Furthermore, by forming S_1 we can group $\widehat{G}_d^{\mathrm{T}}\Lambda\widehat{G}_d x$ operations, and forming S_2 avoids a repeated computation of matrix–vector products with $\widehat{G}_d$.

By Proposition 3.2, $W\Lambda_{12}^{-1}W^{\mathrm{T}}$ is diagonal of size $m \times m$; it takes ℓ operations to compute each element, and hence $\ell m = n$ operations suffice to compute $W\Lambda_{12}^{-1}W^{\mathrm{T}}$. $\widehat{G}_d$ is much larger than $\widehat{G}_{u,-a}$ or $\widehat{G}_{u,a}$. Thus, the most computationally intensive component of this step is the computation of $\widehat{G}_d^{\mathrm{T}}(\Lambda_{12} - (\Lambda_2 - \Lambda_1)^2\Lambda_{12}^{-1})\widehat{G}_d$. The Λ's are diagonal, so calculating $\Lambda_{12} - (\Lambda_2 - \Lambda_1)^2\Lambda_{12}^{-1}$ only takes $O(n)$ operations. Each column of $\widehat{G}_d$ only has a fraction of ρ of its elements non-zero. Moreover, each column only has overlapping non-zeros with approximately $k\rho$ columns. Hence, for each row of $\widehat{G}_d^{\mathrm{T}}(\Lambda_{12} - (\Lambda_2 - \Lambda_1)^2\Lambda_{12}^{-1})\widehat{G}_d$, there are only $k\rho$ non-zeros and each non-zero entry takes $n\rho$ operations to compute. Thus, this step takes $O(k^2 n\rho^2)$ operations.

Next, we look at the computation of $(V\Lambda_{12}^{-1}M_2)^{\mathrm{T}}B^{-1}(V\Lambda_{12}^{-1}M_2)$. We substitute for M_2 using Equation (17) and obtain

$$V\Lambda_{12}^{-1}M_2 = (V\Lambda_{12}^{-1}(\Lambda_2 - \Lambda_1))\widehat{G}_d + (V\Lambda_{12}^{-1}W^{\mathrm{T}})(\Lambda_3\widehat{G}_{u,-a} - \Lambda_4\widehat{G}_{u,a}) \tag{26}$$

Because $\widehat{G}_d$ is much larger than either $\widehat{G}_{u,-a}$ or $\widehat{G}_{u,a}$, the computational time for this step is dominated by the time it takes to compute $(V^{\mathrm{T}}\Lambda_{12}^{-1}(\Lambda_2 - \Lambda_1))\widehat{G}_d$. Obviously, V has the same structure of W, and hence has exactly n non-zero elements. Multiplication of V by a diagonal matrix only takes n operations. Similarly, each column of $\widehat{G}_d$ has roughly $n\rho$ non-zero elements. Thus, multiplying V by any column of $\widehat{G}_d$ takes roughly $O(n\rho)$ operations. Matrix $\widehat{G}_d$ has $k+1$ columns, thus, the multiplication of $V\Lambda_{12}^{-1}(\Lambda_2 - \Lambda_1)$ by $\widehat{G}_d$ takes $O(nk\rho)$ operations. Therefore, computing $V\Lambda_{12}^{-1}M_2$ takes $O(nk\rho)$ operations as well.

Because $V\Lambda_{12}^{-1}M_2$ is $m\times(k+1)$ and B is diagonal, $(V\Lambda_{12}^{-1}M_2)^{\mathrm{T}}B^{-1}(V\Lambda_{12}^{-1}M_2)$ can be computed within $O(k^2 m)$ operations after the computation of $V\Lambda_{12}^{-1}M_2$ is completed as described above. Thus, the total effort for computing $(V\Lambda_{12}^{-1}M_2)^{\mathrm{T}}B^{-1}(V\Lambda_{12}^{-1}M_2)$ is $O(nk\rho + k^2 m)$. In summary, we have:

1. Calculating B: this can be done with $O(ml) = O(n)$ operations.
2. Calculating $M_3 - M_2^{\mathrm{T}}\Lambda_{12}^{-1}M_2$: this requires $O(nk^2\rho^2)$ operations.
3. Calculating $(V\Lambda_{12}^{-1}M_2)^{\mathrm{T}}B^{-1}(V\Lambda_{12}^{-1}M_2)$: this can be done with $O(nk\rho + k^2 m)$ operations.

Thus, the total time to form the Schur complement is $O(nk + nk^2\rho^2 + k^2 m)$.

Once the Schur complement has been formed, the resulting system can be solved in $O(k^3)$ operations using the Cholesky decomposition. This gives a total time of $O(nk + nk^2\rho^2 + k^2 m + k^3)$. Depending on the design parameters, any of these terms can be the dominant one; in other words, either the Schur or Cholesky step can be the critical step.

In the steps presented above, we carefully avoid producing large dense intermediate matrices. For example, M_2 is large and dense. In step 3, instead of forming M_2, we compute $V\Lambda_{12}^{-1}M_2$ directly from the sub-matrices of the constraint matrix A. The largest matrix with the greatest number of non-zeros in the algorithm is $\widehat{G}_d$, which has approximately $2nk\rho$ non-zeros. Thus, using the Cholesky decomposition of S, the amount of memory required by this algorithm is $O(nk\rho)$.

For the sake of comparison, consider computing the Cholesky decomposition of the original normal equations. This would require first computing the normal equation matrix, $A^{\mathrm{T}}\Lambda A$. This is equivalent to computing matrices M_1, M_2, and M_3 from Equation (17). Matrix M_1 has $n\ell$ non-zero elements with $n + n/\ell$ distinct values. These values can be computed in $O(n)$ time using Proposition 3.1. Matrix M_2 can be computed directly from its definition in Equation (17). Noting that W has exactly one non-zero element per column, M_2 is relatively sparse with roughly $2nk\rho$ non-zero elements; it can be computed in $O(nk\rho)$ time. The matrix M_3 is dense and can be computed in $O(nk^2\rho)$ operations. It is straightforward to confirm that computing the right-hand side does not add to the overall complexity. Computing M_3 is the dominant step; thus, the normal equations can be formed in $O(nk^2\rho)$ time.

We now consider the Cholesky decomposition of the normal equations. As the decomposition progresses, it creates a band of non-zeros of width ℓ centered on the principle diagonal, and another band of width $2(k+1)$ along the M_2^{T} matrix—this portion of the matrix fills quite quickly when using the Cholesky algorithm. Therefore, each step of the Cholesky algorithm has $O((k+\ell)^2)$ operations, and there are $n+k$ such steps total. Likewise, the memory requirement is $O(n(k+\ell))$. We will assume that $\ell<k<n$ which holds in typical applications and underestimates the cost of the Cholesky approach otherwise. We now have that applying the Cholesky decomposition to the original normal equations requires $O(nk^2)$ time and $O(nk)$ memory. Thus, the Cholesky step dominates the time and memory requirements if Cholesky decomposition is applied to original normal equations.

In contrast, our approach requires $O(k^2(n\rho^2+k)+nk)$ time and $O(nk\rho)$ memory. By using the Sherman–Morrison–Woodbury formula, our method exploits the sparsity of the G matrices, avoiding the fill that occurs if the Cholesky decomposition (or similar algorithms) is applied to the original normal equations. Thus, our method reduces the memory requirement by a factor of ρ and the time requirement by a factor of $\min(\rho^2, 1/k, k/n)$, depending on the specifics of the design parameters.

3.3. Stability

In general, the Sherman–Morrison–Woodbury formula may be unstable [23]. Here, it is not a concern since we use it primarily for analytical purposes, to derive a factorization-free formula. By [24], factors that affect the stability are the condition number of the original (unperturbed) matrix, the scaling of the low-rank perturbations and the inversion of the small matrix whose size is equal to the rank of the perturbation.

Let $M_1^{(j)}$ denote the jth $\ell\times\ell$ block of M_1. From Equation (17) and Propositions 3.1 and 3.2, we have

$$M_1^{(j)} = D^{(j)} + E^{(j)}$$

$$D^{(j)} = \mathrm{diag}((\Lambda_1)_{(j-1)\ell+1\ldots j\ell} + (\Lambda_2)_{(j-1)\ell+1\ldots j\ell})$$

$$E^{(j)} = ((\Lambda_3)_j + (\Lambda_4)_j)\mathbf{1}_{\ell\times\ell}$$

Note that D is a positive-definite, diagonal matrix, and E is a positive, rank-1 matrix, where all elements have the same value. Let $\{\sigma_i^{(j)}\}$ denote the eigenvalues of this $M^{(j)}$, and suppose $\sigma_m^{(j)}$ and $\sigma_M^{(j)}$ are its minimal and maximal eigenvalues, respectively. Recalling that $\{(\Lambda_{12})_i\}$ denote the diagonal values of Λ_{12}, from the interlacing theorem [25, 26] it follows that $\sigma_m^{(j)} \geqslant \min_i(\Lambda_{12})_i \geqslant 0$.

On the other hand, again using interlacing and by Proposition 3.2 it follows that $\sigma_M^{(j)} \leqslant \max_i (\Lambda_{12})_i + (\Lambda_{34})_j$. Combining all blocks, and taking maximum and minimum over all $\sigma_M^{(j)}$ and $\sigma_m^{(j)}$, it follows that

$$\kappa_2(M_1) \leqslant \kappa_2(\Lambda_{12}) + \kappa_2(\Lambda_{34}) \leqslant \kappa_2(\Lambda)$$

As the linear program solver proceeds towards the solution, Λ becomes increasingly ill conditioned. This is an inherent-property of the interior-point algorithm itself, and from the above it follows that the (1, 1) block, which is used for forming the Schur complement, has a condition number bounded in terms of that of Λ. The small matrix that needs to be inverted in the course of computing M_1^{-1} is B defined in (21). But by Proposition 3.3, B is diagonal, and hence it can be inverted exactly without a concern of catastrophic amplification. The magnitude of the entries of B depend linearly on Λ, hence any conditioning issues are inherent in the problem and are not enhanced by the numerical algorithm.

4. NUMERICAL EXPERIMENTS

We implemented Mehrotra's interior-point linear program solver [16, Chapter 14; 18] using our linear system solver as described in the previous section. Our implementation is in MATLAB. Table I(a) shows the size of the LPs, the number of iterations and time per iteration for buses of various sizes. We used HSPICE [27] to derive bus models and their bit-response functions. All filters have an impulse response of length 4 (i.e. $n_{\text{fir}} = 4$), and the filter for each wire considers the value sent on that wire and its four nearest neighbors in each direction.

Table I. Filter designs for buses of various sizes with $n_{\text{bus}} = 10$ and $k_{\text{mask}} = 8$.

w_{bus}	Number of variables	Number of constraints	Number of LP iterations	Time/iteration (s)	Time/iteration/constraint (ms)
(a)					
4	2393	4752	18	0.98	0.214
8	9529	18 976	20	1.74	0.092
16	38 009	75 840	24	7.20	0.095
32	141 801	303 232	17	30.41	0.100
64	606 713	1 212 672	13	137.84	0.114

w_{bus}	B	$M_3 - M_2^{\text{T}}\Lambda_{12}^{-1}M_2$	$(V\Lambda_{12}^{-1}M_2)^{\text{T}}B^{-1}(V\Lambda_{12}^{-1}M_2)$	Cholesky
(b)				
4	<0.01	0.04	0.01	<0.01
8	0.01	0.12	0.05	<0.01
16	0.07	0.55	0.34	0.02
32	0.29	2.49	2.40	0.08
64	1.21	9.28	14.11	0.56

Note: Panel a shows the performance of the interior-point solver. Panel b shows the time breakdown for critical steps of the linear system solver for a single linear system solve. All numbers reported are in seconds. These times are for a 900 MHz, UltraSparc III Processor.

When the width of the bus doubles, the number of variables and constraints in the LP both go up by a factor of 4. Interestingly, for the larger problems, $w_{\text{bus}} = 32, 64$, we observe a decrease in the number of LP iterations. We contribute this to the stopping criterion we have used, whereby the duality gap is normalized by the size of the problem.

The time per iteration also goes up by approximately a factor of 4, better than the factor of 8 predicted by the asymptotic analysis above. We note that the examples given in Table I are not large in terms of w_{bus} and may not fully reflect the asymptotic regime. We also observe that the time per iterations per number of constraints scales very well with the size of the problem; see last column of the table.

To further analyze the cost of the various components of the algorithm, we list the time breakdown for the critical steps of the linear system solver in Table I(b). The asymptotic behavior of each critical step validates the asymptotic analysis. For example, the time for calculating $M_3 - M_2^{\mathrm{T}} \Lambda_{12}^{-1} M_2$ grows quadratically with the width of the bus, as predicted by the analysis. The time for calculating $(V\Lambda_{12}^{-1}M_2)^{\mathrm{T}} B^{-1} (V\Lambda_{12}^{-1}M_2)$ grows by approximately a factor of 6, slightly better than the factor of 8 predicted by the analysis.

In Figure 6, we show the conditioning of the Schur complement matrix, computed using our algorithm. As is evident from the figure, the matrix has a condition number that deteriorates throughout the iteration, as expected [15], and behaves similar to the condition number of the unreduced normal equations matrix. It is analytically known that the condition number of the Schur complement is bounded from above by that of the unreduced matrix, since the (1, 1) block is symmetric positive definite, as is the unreduced matrix [28, Lemma 3.12].

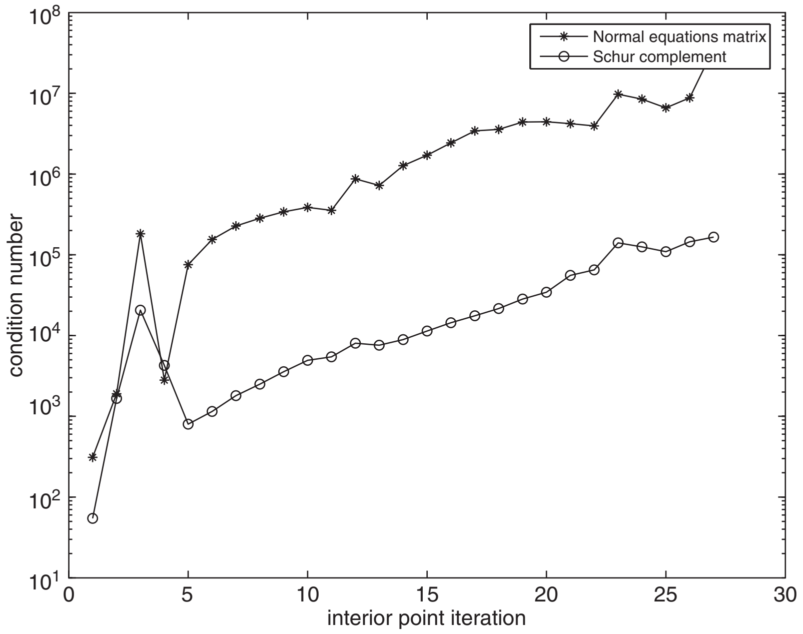

Figure 6. Condition numbers of the matrix of the normal equations and the Schur complement as computed using our algorithm, throughout the LP iteration.

5. CONCLUDING REMARKS

We have presented an efficient and robust approach to solving l_∞ optimal deconvolution. While this deconvolution has a natural linear programming formulation, the resulting linear programs can be quite large. By using an interior-point method to solve these linear programs, the solution of the normal equations at each iteration becomes the critical step for obtaining performance and stability. We have shown that the structure of the matrices involved can be exploited in a way that allows the Schur complement of the normal equations to be formed using a factorization-free formula for the inverse of the (1, 1) block. Furthermore, our approach does not adversely affect the conditioning of the system.

Our interest in l_∞ optimization was motivated by problems that arise in synthesizing equalization filters for high-speed digital signalling. Such problems are naturally formulated as two-dimensional deconvolutions where the l_∞ norm corresponds to the discrete quantization of digital systems. Our linear system solver makes l_∞ optimization a tractable alternative to least-squares methods.

Our solver exploits the special structure of the W matrix (see Equation (9)) that arises because of the duality between the l_∞ and l_1 norms. The l_∞ formulation shown in Equation (6) assumes that the system is time-invariant but not space invariant; in other words, different lines in the system have different characteristics, while each line itself is time invariant. We also exploited the sparsity of the G matrix that computes the channel responses to unit inputs as a function of the filter coefficients. This sparsity arises from the assumption that the spatial dimension is not shift invariant. As a consequence, the deconvolver has different coefficients for each line input to the filter. Each row of G corresponds to one such line, and has zeros in the columns corresponding to deconvolver coefficients for the other lines. Thus, this sparsity should be available when applying our approach to other l_∞ deconvolution problems.

If the system is also space invariant, i.e. lines are identical, the problem reduces to optimization for a single line. In this case, the LP still has the same structure but with much smaller size: k_{disturb} becomes linear in w_{bus}. Furthermore, the convolution matrices become block Toeplitz or circulant [29], and matrix multiplications can be performed efficiently using FFT methods. This might allow further efficiency in the linear system solver. In contrast, when both dimensions are not shift invariant, for example, in image processing applications, the same structure arises, but with a much larger size: k_{disturb} becomes quadratic in both dimensions. The formulation that we have presented in this paper still applies and should, in principle, lead to efficient solutions.

While equalization filter synthesis for high-speed buses is naturally expressed as a two-dimensional deconvolution, higher dimensional, l_∞ deconvolution problems give rise to the same matrix structure. Our methods could be employed for those problems as well. Consider a d-dimensional deconvolution problem where the channel to be deconvolved is shift invariant in dimensions $1 \ldots \tilde{d}$, and non-shift-invariant in dimensions $\tilde{d}+1 \ldots d$. Let w_i be the length of the convolution kernel for the channel for dimension i, and let k denote the total number of filter coefficients. Then, the linear program for synthesizing an l_∞ optimal FIR filter for deconvolution has roughly $n = (\prod_{i=1}^{\tilde{d}} w_i)(\prod_{i=\tilde{d}+1}^{d} w_i^2)$ variables (the disturbance terms) and twice that many constraints (the absolute value computations). The matrix for the normal equations is $(n+k+1) \times (n+k+1)$, and the upper-left $n \times n$ submatrix has the block diagonal structure that is required for our construction. Thus, we can reduce this to a $(k+1) \times (k+1)$ dense system that can be solved directly. As with the two-dimensional case, the higher-dimensional generalization of our approach is likely to be computationally efficient and stable.

REFERENCES

1. Dally WJ, Poulton JW. *Digital Systems Engineering*. Cambridge University Press: Cambridge, MA, 1998.
2. Chiang P, Dally WJ *et al.* 20-gb/s 0.13-m CMOS serial link transmitter using an LC-PLL to directly drive the output multiplexer. *IEEE Journal of Solid-state Circuits* 2005; **40**(4):1004–1011.
3. Zerbe JL, Chau RS *et al.* A 2Gb/s/pin 4-PAM parallel bus interface with transmit crosstalk cancellation, equalization and integrating receivers. *IEEE International Solid State Circuits Conference*, San Franscisco, CA, 2001; 430–432.
4. Crespo PM, Honig ML. Pole-zero decision feedback equalization with a rapidly converging adaptive IIR algorithm. *IEEE Journal of Selected Areas in Communications* 1991; **9**:817–829.
5. Barry JR, Lee EA, Messerschmitt DG. *Digital Communication*. Kluwer Academic Publishers: Dordrecht, 2004.
6. Dally WJ, Poulton JW. Transmitter equalization for 4-GBPs signaling. *IEEE Micro* 1997; **1**:48–56.
7. Stojanovic V, Ho A, Garlepp B, Chen F, Wei J. Autonomous dual-mode (PAM2/4) serial link transceiver with adaptive equalization and data recovery. *IEEE Journal of Solid-state Circuits* 2005; **April**:1012–1025.
8. Ren J, Greenstreet MR. Crosstalk cancellation for realistic PCB buses. *Proceedings of the 14th International Workshop on Power and Timing Modeling, Optimization and Synthesis* (*PATMOS*). Lecture Notes in Computer Science. Springer: Berlin, 2004.
9. Andersen KD. A modified Schur-complement method for handling dense columns in interior-point methods for linear programming. *ACM Transactions on Mathematical Software* 1996; **22**(3):348–356.
10. Grigoriadis MD, Khachiyan LG. An interior point method for bordered block-diagonal linear programs. *SIAM Journal on Optimization* 1996; **6**(4):913–932.
11. Choi IC, Monma CL, Shanno DF. Further development of a primal–dual interior point method. *ORSA Journal on Computing* 1990; **2**:304–311.
12. Ren J, Greenstreet MR. A unified optimization framework for equalization filter synthesis. *Proceedings of the 42nd ACM/IEEE Design Automation Conference*, Anaheim, CA, June 2005; 638–643.
13. Ren J. Equalizing filter design for high-speed off-chip buses. *Ph.D. Thesis*, University of British Columbia, Department of Computer Science, 2005.
14. Oppenheim AV, Schafer RW, Buck JR. *Discrete-time Signal Processing* (2nd edn). Prentice-Hall: Englewood Cliffs, NJ, 1999.
15. Forsgren A, Gill PE, Wright MH. Interior methods for nonlinear optimization. *SIAM Review* 2002; **44**(4):525–597.
16. Nocedal J, Wright SJ. *Numerical Optimization*. Springer Series in Operations Research. Springer: Berlin, 1999.
17. Wright SJ. *Primal—dual Interior-point Methods*. SIAM: Philadelphia, PA, 1997.
18. Mehrotra S. On the implementation of a primal–dual interior point method. *SIAM Journal on Optimization* 1992; **2**:575–601.
19. Lustig IJ, Marsten RE, Shanno DF. On implementing Mehrotra's predictor–corrector interior-point method for linear programming. *SIAM Journal on Optimization* 1992; **2**(3):435–449.
20. Zhang Y. Solving large-scale linear programs by interior-point methods under the MATLAB environment. *Technical Report TR96-01*, University of Maryland, July 1995.
21. Trottenberg U, Oosterlee C, Schüller A. *Multigrid*. Academic Press: New York, 2001.
22. Henderson HV, Searle SR. On deriving the inverse of a sum of matrices. *SIAM Review* 1981; **23**(1):53–60.
23. Higham NJ. *Accuracy and Stability of Numerical Algorithms* (2nd edn). SIAM: Philadelphia, PA, 2002.
24. Yip EL. A note on the stability of solving a rank-p modification of a linear system by the Sherman–Morrison–Woodbury formula. *SIAM Journal on Scientific and Statistical Computations* 1986; **7**(3):507–513.
25. Golub GH, Van Loan CF. *Matrix Computations* (3rd edn). John Hopkins University Press: Baltimore, MD, 1996.
26. Wilkinson JH. *The Algebraic Eigenvalue Problem*. Clarendon Press: Oxford, 1965.
27. Avant! Software. *Star-Hspice Manual*, *Release 2001.4*, Avant! Software, 2001.
28. Axelsson O. *Iterative Solution Methods*. Cambridge University Press: Cambridge, MA, 1994.
29. Ren J, Greenstreet MR. Synthesizing optimal filters for crosstalk-cancellation for high-speed buses. *Proceedings of the 40th ACM/IEEE Design Automation Conference*, San Deigo, CA, June 2003; 592–597.