Efficient Computation of Sparse Elements of the Inverse of a Sparse Near-Tridiagonal Matrix with Application to the Nerve Equation

Anthony M. Zador
Interdepartmental Neuroscience Program
Yale University School of Medicine
367 Cedar Street
New Haven, CT 06520
zador@yale.edu

Barak A. Pearlmutter*
Dept. of Computer Sci. and Eng.
Oregon Graduate Institute
19600 NW von Neumann Drive
Beaverton, OR 97006-1999
bap@cse.ogi.edu

Abstract

Standard algorithms for computing the inverse of a tridiagonal matrix (or more generally, any Hines matrix) compute the entire inverse, which is not sparse. For some problems, only the elements of the inverse at locations corresponding to nonzero elements in the original matrix are required. We present an algorithm that efficiently computes only these elements in $O(n)$ time and memory. This algorithm is useful in solving discretized systems of partial differential equations that arise when computing electrical flow along a branched structure, such as a neuron's dendritic arbor.

1 Introduction

The electrical parameters and connectivity in branched RC networks define a sparse matrix B which has nonzero elements only at locations that correspond to electrical connections. This sparseness can be exploited to compute efficiently the distribution of potential in such networks (Hines, 1984; Mascagni, 1989). Some applications, however, require the transfer impedance matrix, $K = B^{-1}$, (Carnevale and Johnston, 1982; Koch et al., 1982), which permits the direct computation of the potential at any point j due to an input at any other point i. For example, a simple function of the impedance matrix can be used to visualize directly the electrical properties of a branched structure via the *morphoelectrotonic transform* (Zador, 1992). Although K (unlike B) is a full matrix, in computing the morphoelectrotonic transform only the elements of K at locations corresponding to nonzero elements of B are required. Moreover, the remaining elements of K can be trivially computed from this sparse subset. In this paper we present an efficient method for calculating just the requisite elements of K.

This paper is organized into five sections. In section 2 we provide a formal statement of the matrix inversion problem. In section 3 we present a two-sided variant of Gaussian elimination that computes only the desired sparse subset of K, without devoting any extraneous computation to the calculation of the rest of K, and illustrate the algorithm with an example. In section 4 we show how the problem arises in the computation of the morphoelectrotonic transform, and in section 5 we discuss extensions and limitations of the algorithm.

* To whom all correspondence should be addressed

2 Problem statement

Given a sparse matrix B whose structure corresponds to an acyclic graph, we want to compute only those elements K_{ij} of $K = B^{-1}$ where B_{ij} is nonzero.[1] Our procedure for accomplishing this applies to a particular class of sparse matrices B, which we call *Hines* matrices. As we will show below, the connectivity of any acyclic graph can be represented as a Hines matrix by appropriate numbering of the vertices. We therefore proceed with the formal definition of a Hines matrix.

Definition: a matrix B is a *Hines* matrix when

1. The diagonal elements B_{jj} are nonzero.
2. B_{ij} is nonzero iff B_{ji} is nonzero.
3. For any nonzero B_{ij} with $i < j$, there is no h such that $h > j$ and B_{ih} is nonzero.

The second condition requires B to be structurally (although not numerically) symmetric, and the third requires that B have no more than one nonzero element to the right of the diagonal in any row, and by structural symmetry, no more than one nonzero element below the diagonal in any column. This condition is implicit in the Hines (1984) algorithm.

3 Algorithm

The algorithm consists of three parts. First a Hines matrix B is generated from an acyclic graph by appropriate numbering of the vertices. Next, in a forward elimination pass, row and column operations are performed on B that reduce it to the identity matrix. Finally, in a backward pass, the duals of these operations are applied in reverse order to an identity matrix, resulting in B^{-1}. The key to the algorithm's efficiency is the fact that, in this backward pass, only the sparse subset of B^{-1} needs to be retained.

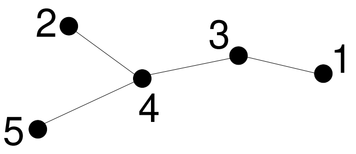
3.1 Numbering the nodes of an acyclic graph so the connectivity matrix is in Hines form

The example in figure 1 shows the correspondence between a graph and its connectivity matrix. There is a nonzero element at the intersection of row i and column j when there is a link from vertex i to vertex j; in addition, the diagonal is nonzero. The following procedure numbers the vertices of an acyclic graph G in such a way that its connection matrix is in Hines form.

int $i = 1$;

while (there remain unlabeled vertices in G) {
  Choose an unlabeled vertex g linked to at most one other unlabeled vertex;
  Label g with the number i;
  $i = i + 1$;
}

[1] When we say that an element is nonzero, we mean it is an element that is part of the sparse set of elements that may but need not be nonzero.



$$\begin{pmatrix} 1 & & \times & & \\ & 2 & & \times & \\ \times & & 3 & \times & \\ & \times & \times & 4 & \times \\ & & & \times & 5 \end{pmatrix}$$

Figure 1: A graph labeled so that its connectivity matrix is in Hines form.

Note that there can be many candidates for the choice of g in each cycle through the loop. The vertex ordering portion of the Hines (1984) algorithm is a special case of this algorithm in which g is chosen in a particular depth-first local order.

3.2 Two-sided Gaussian elimination

The algorithm is conveniently expressed as a sequence of left and right matrix multiplications corresponding to row and column operations,

$$L^{(n)}(\cdots(L^{(2)}(L^{(1)}BR^{(1)})R^{(2)})\cdots)R^{(n)} = I \tag{1}$$

where B is the nearly-tridiagonal $n \times n$ Hines matrix defined above, $L^{(1)}, \ldots, L^{(n)}$ and $R^{(1)}, \ldots, R^{(n)}$ are row and column operations defined below, and I is the identity matrix. Using $L = L^{(n)} \cdots L^{(1)}$ and $R = R^{(1)} \cdots R^{(n)}$, we can express (1) as $B = L^{-1}IR^{-1}$, so

$$B^{-1} = RIL = R^{(1)}(\cdots(R^{(n-1)}(R^{(n)}IL^{(n)})L^{(n-1)})\cdots)L^{(1)}. \tag{2}$$

The j^{th} matrix $L^{(j)}$ is defined so that it eliminates the nonzero element of B below the diagonal element B_{jj}, and similarly $R^{(j)}$ eliminates the element to the right of B_{jj}. Specifically, if we define the j^{th} partial product $B^{(j)} = L^{(j)}B^{(j-1)}R^{(j)}$, with $B^{(0)} = B$, then the operation matrices are nearly identity matrices, but with one modified diagonal element that serves to normalize $B^{(j)}_{jj}$,

$$L^{(j)}_{jj} = R^{(j)}_{jj} = \frac{1}{\sqrt{B^{(j-1)}_{jj}}}, \tag{3}$$

and one off-diagonal element,

$$L^{(j)}_{hj} = -\frac{B^{(j-1)}_{hj}}{B^{(j-1)}_{jj}} \qquad\qquad R^{(j)}_{jh} = -\frac{B^{(j-1)}_{jh}}{B^{(j-1)}_{jj}} \tag{4}$$

where h gives the column of the nonzero element to the right of the diagonal in the j^{th} row. Since B is in Hines form there is only one such column h for each row j, and since B is structurally symmetric, h also gives the row of the nonzero element below the diagonal element B_{jj}.

The last matrices, $L^{(n)}$ and $R^{(n)}$ have no nonzero off-diagonal elements, and serve only to normalize the last element of the diagonal. All n forward matrices must be determined before the backward computation

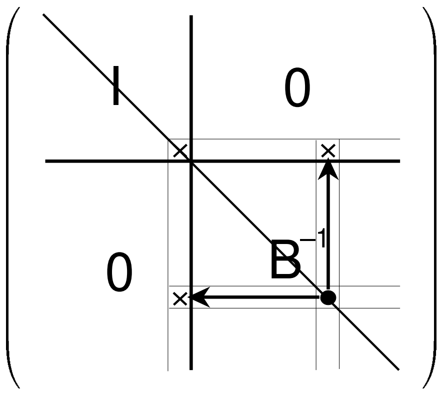


Figure 2: Diagrammatic proof that elements outside the sparse set can be safely ignored in the backward phase of the calculation of the sparse subset elements of B^{-1}.

of B^{-1} begins, because their duals must be applied in reverse order to the identity matrix, as indicated in (2). Note that when B is symmetric, as in the example below, $L^{(j)}$ is the transpose of $R^{(j)}$.

For the forward pass it is clear by inspection that no spurious elements are created. For the backward pass, although elements outside the sparse subset we are interested in are created at each step of the algorithm, none of them enter into the calculation of the values of any future structural elements, because of the Hines condition. This is illustrated in figure 2, where at each step in the backward pass, the lower right hand block of the matrix contains the portion of B^{-1} computed thus far, while the remainder of the matrix is 0 except for ones along the diagonal. At each step, the lower right hand block is expanded by one row and one column. These operations are dual to the operations used to cancel out the off-diagonal elements, whose locations are denoted by crosses. The dual operations instead create these same off-diagonal elements, and also fill all the other elements to the right of and below the diagonal in the row and column in question. However, because of the Hines condition, we are concerned with only one element in the row and column, and that element depends solely on the diagonal element denoted by a dot, and on the coefficients of the row and column operations.

We illustrate the algorithm in figures 4 and 5 with a simple numerical example on a 4×4 matrix. In this example, only the structurally nonzero elements are retained. Elements whose values are not computed, but are nonzero, are indicated with dots. In the first step, right and left multiplications zero the first row and column, and normalize the first diagonal element to 1. In the process, a lower diagonal element is modified. In the second step, the off-diagonal elements in the second row and column are set to zero, the second diagonal element is normalized to 1, and again, a lower diagonal element is modified. This process proceeds until the entire matrix has been converted to the identity matrix. The row and column operations used in this process are stored, and their duals are used, in reverse order, in the backward pass. For instance, if in the forward pass the first row operator adds 1/3 of row 1 to row 4 and then multiplies row 1 by $1/\sqrt{5}$, then in the backward pass the last column operation multiplies column 1 by $1/\sqrt{5}$ and then adds 1/3 of column 4 to column 1.

In the optimized implementation shown in figure 3, only the elements $B_{jj}^{(j)}$ (and the off-diagonal elements of B) are retained for the backward pass. The $L^{(j)}$ and $R^{(j)}$ operators are not retained, as they can be trivially computed from $B_{jj}^{(j)}$ and the off-diagonal elements of B. As a result, no square roots are required, and if B is real then no complex intermediate values or complex arithmetic is necessary. If space is at a premium, a further optimization (not shown) can reduce the storage requirements by gradually overwriting B with K in the backward pass.

4 Application to the nerve equation

Two-sided Gaussian elimination can be applied to the numerical computation of the nerve equation. Electrical flow in passive neurons with branched dendritic trees can be described by a system of partial differential equations (Jack et al., 1983). Specifically the potential along each branch is given by the cable equation

$$\frac{d}{2R}\frac{\partial^2 V}{\partial x^2} = C\frac{\partial V}{\partial t} + GV + J$$

where $V(x,t)$ is the potential along each branch, t is time, x is space, $J(x,t)$ is an electrical current source, and $C(x)$, $R(x)$, $G(x)$ and $d(x)$ are parameters describing the electrical and physical properties of the neuronal membrane. Spatially discretizing this system results in the matrix equation

$$B'V = C\dot{V} + GV + J$$

where B' is a Hines matrix and C, G and J are tridiagonal. In particular, B' is tridiagonal along each branch, with offdiagonal elements corresponding to segments connected at branchpoints.

Taking the Fourier transform of each side, we have $B(\omega)V(\omega) = J(\omega)$ where we have defined $B(\omega) = B' - (G - i\omega C)$. Defining the transfer impedance matrix $K(\omega) = B(\omega)^{-1}$, we obtain

$$V(\omega) = K(\omega)\ J(\omega).$$

The morphoelectrotonic transform provides a graphical method for appreciating the electrotonic structure of a neuron's complex branched dendritic tree. In the morphoelectrotonic transform, the voltage log-attenuation $L_{ih} = \log(K_{ii}/K_{ih})$ between two adjacent points i and h replaces the physical distance in graphical representations of the neuron. The advantage of this representation is that the log-attenuation is additive—for j between i and h, $L_{ih} = L_{ij} + L_{jh}$—so that the electrical coupling between any two points in a neuron can be computed directly from the sparse subset stored.

5 Extensions and Limitations

The most serious problem with the algorithm presented in this paper is that pivoting is not feasible, as pivoting does not preserve the Hines condition. This means that the stability advantages of pivoting are sacrificed. For the physical systems in which we are interested, the matrix B is well conditioned, so the inability to pivot is not a serious limitation.

In addition to the application to the nerve equation, this technique may prove useful for computing the frequency-dependent voltage attenuation and phase lag in passive VLSI clock distribution networks. A simple augmentation of the algorithm can provide the sensitivity of the electrotonic distance and phase lag to the electrical parameters of the various components, which might prove helpful in optimizing such circuits.

Acknowledgments

First and foremost, we would like to thank Michael Hines for making his NEURON simulator publicly available. The computational neuroscience community owes much to his generosity, and to his putting the good of science above the prospect of financial gain. BAP was partially supported by grants NSF ECS-9114333 and ONR N00014-92-J-4062 to John Moody, and we thank Christoph Koch for the generous contribution of computational facilities.

References

Carnevale, N. T. and Johnston, D. (1982). Electrophysiological characterization of remote chemical synapses. *Journal of Neurophysiology*, 47:606–621.

Hines, M. (1984). Efficient computation of branched nerve equations. *International Journal of Biomedical Computing*, 15:69–76.

Jack, J., Noble, A., and Tsien, R. W. (1983). *Electrical Current Flow in Excitable Membranes*. Oxford, second edition.

Koch, C., Poggio, T., and Torre, V. (1982). Retinal ganglion cells: A functional interpretation of dendritic morphology. *Proceedings of the Royal Society of London B*, 298:227–264.

Mascagni, M. V. (1989). Numerical methods for neuronal modeling. In Koch, C. and Segev, I., editors, *Methods in neuronal modeling: From synapses to networks*, pages 439–484. MIT Press.

Zador, A. M. (1992). *Biophysics of Computation in Single Hippocampal Neurons*. PhD thesis, Yale University, Interdepartmental Neuroscience Program.

```
int h[n-1];            /* $h[j]$ is the column of the nonzero element to the right of $B_{jj}$ */

/* The array $B$ is not modified by the code.  Sparse elements represented thus: */
float Bd[n];           /* Bd[j] = $B_{jj}$ */
float Br[n-1];         /* Br[j] = $B_{j,h[j]}$ */
float Bl[n-1];         /* Bl[j] = $B_{h[j],j}$ */

/* This sparse subset of $K = B^{-1}$ is the output.  Represented like $B$: */
float Kd[n], Kr[n-1], Kl[n-1];

/* Retains information computed in the forward pass and needed later. */
float BJd[n];          /* BJd[j] = $B^{(j)}_{jj}$ */

/* Don't retain $R^{(j)}$ or $L^{(j)}$; they can be computed from BJd, Bl, and Br. */

/* Initialize $B^{(j)}_{jj} = 1$. */
for(j = 1 ; j <= n ; j++)
  BJd[j] = 1;

/* Forward elimination pass over $B$: */
for(j = 1 ; j <= n-1 ; j++)
   BJd[h[j]] -= Br[j] * Bl[j] / BJd[j];

/* Backward pass to compute $K$: */
Kd[n] = 1/BJd[n];
for(j = n-1 ; j >= 1 ; j--) {
   Kd[j] = 1/BJd[j] + ( Kd[h[j]] * Bl[j] * Br[j] ) / ( BJd[j] * BJd[j] );
   Kr[j] = - Kd[h[j]] * Br[j] / BJd[j];
   Kl[j] = - Kd[h[j]] * Bl[j] / BJd[j];
}
```

Figure 3: Code for the forward elimination and backward pass, written for an asymmetric but real B. When B is symmetric (due to boundary conditions it is not for the nerve equation) then Bl = Br and Kl = Kr. If B is complex, as in the nerve equation, complex arithmetic becomes necessary throughout.

$$
\underbrace{
\underset{L^{(1)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ \mathbf{\frac{1}{3}} & & 1 & \\ & & & 1 \end{pmatrix}}
\underset{B^{(0)} = B}{\begin{pmatrix} \mathbf{1} & & -\mathbf{\frac{1}{3}} & \\ & 1 & -\frac{1}{5} & \\ -\mathbf{\frac{1}{3}} & -\frac{1}{5} & 1 & -\frac{1}{7} \\ & & -\frac{1}{7} & 1 \end{pmatrix}}
\underset{R^{(1)}}{\begin{pmatrix} 1 & & \mathbf{\frac{1}{3}} & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}}
}
\downarrow
$$

$$
\underbrace{
\underset{L^{(2)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & \mathbf{\frac{1}{5}} & 1 & \\ & & & 1 \end{pmatrix}}
\underset{B^{(1)} = L^{(1)} B^{(0)} R^{(1)}}{\left(\begin{array}{c|ccc} 1 & & 0 & \\ \hline & \mathbf{1} & -\mathbf{\frac{1}{5}} & \\ 0 & -\mathbf{\frac{1}{5}} & \frac{8}{9} & -\frac{1}{7} \\ & & -\frac{1}{7} & 1 \end{array}\right)}
\underset{R^{(2)}}{\begin{pmatrix} 1 & & & \\ & 1 & \mathbf{\frac{1}{5}} & \\ & & 1 & \\ & & & 1 \end{pmatrix}}
}
\downarrow
$$

$$
\underbrace{
\underset{L^{(3)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \sqrt{\mathbf{\frac{225}{191}}} & \\ & & \mathbf{\frac{225}{1337}} & 1 \end{pmatrix}}
\underset{B^{(2)} = L^{(2)} B^{(1)} R^{(2)}}{\left(\begin{array}{cc|cc} 1 & & 0 & \\ & 1 & 0 & \\ \hline 0 & 0 & \mathbf{\frac{191}{225}} & -\mathbf{\frac{1}{7}} \\ & & -\mathbf{\frac{1}{7}} & 1 \end{array}\right)}
\underset{R^{(3)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \sqrt{\mathbf{\frac{225}{191}}} & \mathbf{\frac{225}{1337}} \\ & & & 1 \end{pmatrix}}
}
\downarrow
$$

$$
\underbrace{
\underset{L^{(4)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & \sqrt{\mathbf{\frac{9359}{9134}}} \end{pmatrix}}
\underset{B^{(3)} = L^{(3)} B^{(2)} R^{(3)}}{\left(\begin{array}{ccc|c} 1 & & 0 & \\ & 1 & 0 & \\ 0 & 0 & 1 & 0 \\ \hline & & 0 & \mathbf{\frac{9134}{9359}} \end{array}\right)}
\underset{R^{(4)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & \sqrt{\mathbf{\frac{9359}{9134}}} \end{pmatrix}}
}
\downarrow
$$

$$
\underset{L^{(4)} B^{(3)} R^{(4)} = I}{\begin{pmatrix} 1 & & 0 & \\ & 1 & 0 & \\ 0 & 0 & 1 & 0 \\ & & 0 & 1 \end{pmatrix}}
$$

Figure 4: Forward elimination, see text for details.

$$
\underbrace{
\underset{R^{(4)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & \sqrt{\frac{9359}{9134}} \end{pmatrix}}
\quad
\underset{I}{\left(\begin{array}{cccc|} 1 & & 0 & \\ & 1 & 0 & \\ 0 & 0 & 1 & 0 \\ & & 0 & 1 \\ \hline \end{array}\right)}
\quad
\underset{L^{(4)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & \sqrt{\frac{9359}{9134}} \end{pmatrix}}
}_{\downarrow}
$$

$$
\underbrace{
\underset{R^{(3)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \sqrt{\frac{225}{191}} & \frac{225}{1337} \\ & & & 1 \end{pmatrix}}
\quad
\underset{K^{(4)} = R^{(4)} I L^{(4)}}{\left(\begin{array}{ccc|c} 1 & & 0 & \\ & 1 & 0 & \\ 0 & 0 & 1 & 0 \\ \hline & & 0 & \frac{9359}{9134} \end{array}\right)}
\quad
\underset{L^{(3)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \sqrt{\frac{225}{191}} & \\ & & \frac{225}{1337} & 1 \end{pmatrix}}
}_{\downarrow}
$$

$$
\underbrace{
\underset{R^{(2)}}{\begin{pmatrix} 1 & & & \\ & 1 & \frac{1}{5} & \\ & & 1 & \\ & & & 1 \end{pmatrix}}
\quad
\underset{K^{(3)} = R^{(3)} K^{(4)} L^{(3)}}{\left(\begin{array}{cc|cc} 1 & & 0 & \\ & 1 & 0 & \\ \hline 0 & 0 & \frac{11025}{9134} & \frac{1575}{9134} \\ & & \frac{1575}{9134} & \frac{9359}{9134} \end{array}\right)}
\quad
\underset{L^{(2)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & \frac{1}{5} & 1 & \\ & & & 1 \end{pmatrix}}
}_{\downarrow}
$$

$$
\underbrace{
\underset{R^{(1)}}{\begin{pmatrix} 1 & & \frac{1}{3} & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}}
\quad
\underset{K^{(2)} = R^{(2)} K^{(3)} L^{(2)}}{\left(\begin{array}{c|ccc} 1 & & 0 & \\ \hline & \frac{9575}{9134} & \frac{2205}{9134} & \cdot \\ 0 & \frac{2205}{9134} & \frac{11025}{9134} & \frac{1575}{9134} \\ & \cdot & \frac{1575}{9134} & \frac{9359}{9134} \end{array}\right)}
\quad
\underset{L^{(1)}}{\begin{pmatrix} 1 & & & \\ & 1 & & \\ \frac{1}{3} & & 1 & \\ & & & 1 \end{pmatrix}}
}_{\downarrow}
$$

$$
\left(\begin{array}{|cccc} \hline \frac{10359}{9134} & \cdot & \frac{3675}{9134} & \cdot \\ \cdot & \frac{9575}{9134} & \frac{2205}{9134} & \cdot \\ \frac{3675}{9134} & \frac{2205}{9134} & \frac{11025}{9134} & \frac{1575}{9134} \\ \cdot & \cdot & \frac{1575}{9134} & \frac{9359}{9134} \end{array}\right)
$$

$$
B^{-1} = K^{(1)} = R^{(1)} K^{(2)} L^{(1)}
$$

Figure 5: Backward pass, see text for details.