Fast Stable Solver for Sequentially Semi-separable Linear Systems of Equations

S. Chandrasekaran[1⋆], P. Dewilde[3], M. Gu[2⋆⋆], T. Pals[1∗], and A.-J. van der Veen[3]

[1] University of California, Santa Barbara
shiv@ece.ucsb.edu
[2] University of California, Berkeley
mgu@math.berkeley.edu
[3] DIMES, Delft

1 Introduction

In this paper we will present a fast backward stable algorithm for the solution of certain structured matrices which can be either sparse or dense. It essentially combines the fast solution techniques for banded plus semi-separable linear systems of equations of Chandrasekaran and Gu [4] with similar techniques of Dewilde and van der Veen for time-varying systems [12].

We will also use the proposed techniques to suggest fast direct solvers for a class of spectral methods for which there had been no known fast direct solvers (not even unstable ones). This will illustrate the usefulness of the algorithms presented in this paper. This is the spectral method by Kress [11] for solving the integral equations of classical exterior scattering theory in two dimensions.

To be more specific, let A be an $N \times N$ (possibly complex) matrix satisfying the matrix structure. Then there exist n positive integers $m_1, \cdots, m_n$ with $N = m_1 + \cdots + m_n$ to block-partition A as $A = (A_{i,j})$, where $A_{ij} \in \mathbf{C}^{m_i \times m_j}$ satisfies

$$A_{ij} = \begin{cases} D_i, & \text{if } i = j, \\ U_i W_{i+1} \cdots W_{j-1} V_j^H, & \text{if } j > i, \\ P_i R_{i-1} \cdots R_{j+1} Q_j^H, & \text{if } j < i. \end{cases} \tag{1}$$

Here we use the superscript H to denote the Hermitian transpose. The sequences $\{U_i\}_{i=1}^{n-1}$, $\{V_i\}_{i=2}^{n}$, $\{W_i\}_{i=2}^{n-1}$, $\{P_i\}_{i=2}^{n}$, $\{Q_i\}_{i=1}^{n-1}$, $\{R_i\}_{i=2}^{n-1}$ and $\{D_i\}_{i=1}^{n}$ are all matrices whose dimensions are defined in Table 1. While any matrix can be represented in this form for large enough k_i's and l_i's, our main focus will be on matrices of this special form that have relatively small values for the k_i's and l_i's (see Section 3). In the above equation, empty products are defined to

⋆ Partially supported by grant B521537 from Lawrence Livermore national Laboratory and NSF Career Award CCR-9734290.

⋆⋆ This research was supported in part by NSF Career Award CCR-9702866 and by Alfred Sloan Research Fellowship BR-3720.

S. Sahni et al. (Eds.) HiPC 2002, LNCS 2552, pp. 545–554, 2002.

Table 1. Dimensions of matrices in (1). k_i and l_i are column dimensions of U_i and P_i, respectively

Matrix	U_i	V_i	W_i	P_i	Q_i	R_i
Dimensions	$m_i \times k_i$	$m_i \times k_{i-1}$	$k_{i-1} \times k_i$	$m_i \times l_i$	$m_i \times l_{i+1}$	$l_{i+1} \times l_i$

be the identity matrix. For $n = 4$, the matrix A has the form

$$A = \begin{pmatrix} D_1 & U_1 V_2^H & U_1 W_2 V_3^H & U_1 W_2 W_3 V_4^H \\ P_2 Q_1^H & D_2 & U_2 V_3^H & U_2 W_3 V_4^H \\ P_3 R_2 Q_1^H & P_3 Q_2^H & D_3 & U_3 V_4^H \\ P_4 R_3 R_2 Q_1^H & P_4 R_3 Q_2^H & P_4 Q_3^H & D_4 \end{pmatrix}.$$

We say that the matrix A is **sequentially semi-separable** if it satisfies (1). In the case where all W_i and R_i are identities, A reduces to a block-diagonal plus semi-separable matrix, which can be handled directly using techniques in Chandrasekaran and Gu [4]. It is shown in [12] that this class of matrices is closed under inversion and includes banded matrices, semi-separable matrices as well as their inverses as special cases.

It should be noted that the sequentially semi-separable structure of a given matrix A depends on the sequence m_i. Different sequences will lead to different representations. Through out this paper we will assume that the D_i's are square matrices. The methods in this paper can be generalized to non-square representations too, but that matter will not be pursued here.

2 Fast Backward Stable Solver

In this section we describe a recursive and fast backward stable solver for the linear system of equations $Ax = b$, where A satisfies (1) and b itself is an unstructured matrix

We assume that the sequentially semi-separable matrix A is represented by the seven sequences $\{U_i\}_{i=1}^{n-1}$, $\{V_i\}_{i=2}^{n}$, $\{W_i\}_{i=2}^{n-1}$, $\{P_i\}_{i=2}^{n}$, $\{Q_i\}_{i=1}^{n-1}$, $\{R_i\}_{i=2}^{n-1}$ and $\{D_i\}_{i=1}^{n}$ as in (1). We also partition $x = (\, x_i \,)$ and $b = (\, b_j \,)$ such that x_i and b_i have m_i rows. As in the 4×4 example, there are two cases at each step of the recursion.

Case of $n > 1$ and $k_1 < m_1$: Elimination. Our goal is to do orthogonal eliminations on both sides of A to create an $(m_1 - k_1) \times (m_1 - k_1)$ lower triangular submatrix at the top left corner of A.

We perform orthogonal eliminations by computing QL and LQ factorizations

$$U_1 = q_1 \begin{array}{c} \begin{pmatrix} 0 \\ \hat{U}_1 \end{pmatrix} \end{array} \begin{array}{l} m_1 - k_1 \\ k_1 \end{array} \quad \text{and} \quad \left(q_1^H D_1 \right) = \begin{array}{l} \\ m_1 - k_1 \\ k_1 \end{array} \begin{array}{c} \begin{array}{cc} m_1 - k_1 & k_1 \end{array} \\ \begin{pmatrix} D_{11} & 0 \\ D_{21} & D_{22} \end{pmatrix} \end{array} w_1,$$

where q_1 and w_1 are unitary matrices. To complete the eliminations, we also need to apply q_1^H to b_1 and w_1 to Q_1 to obtain

$$q_1^H b_1 = \begin{array}{l} m_1 - k_1 \\ k_1 \end{array} \begin{pmatrix} \beta_1 \\ \gamma_1 \end{pmatrix} \quad \text{and} \quad w_1 Q_1 = \begin{array}{l} m_1 - k_1 \\ k_1 \end{array} \begin{pmatrix} Q_{11} \\ \hat{Q}_1 \end{pmatrix}.$$

Equations (1) have now become

$$\begin{pmatrix} q_1^H & 0 \\ 0 & I \end{pmatrix} A \begin{pmatrix} w_1^H & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} w_1 & 0 \\ 0 & I \end{pmatrix} x = \begin{pmatrix} q_1^H & 0 \\ 0 & I \end{pmatrix} b - \begin{pmatrix} 0 \\ P_2 \\ P_3 R_2 \\ P_4 R_3 R_2 \\ \vdots \\ P_n R_{n-1} \cdots R_2 \end{pmatrix} \tau, \tag{2}$$

We now orthogonally transform the unknowns x_1 and solve the $(m_1 - k_1) \times (m_1 - k_1)$ lower triangular system of equations. Let $\begin{array}{l} m_1 - k_1 \\ k_1 \end{array} \begin{pmatrix} z_1 \\ \hat{x}_1 \end{pmatrix} = w_1 x_1$. Then the first $m_1 - k_1$ equations of (2) has been simplified to $D_{11} z_1 = \beta_1$. Hence we compute $z_1 = D_1^{-1} \beta_1$ by forward substitution.

We further compute $\hat{b}_1 = \gamma_1 - D_{21} z_1$. This in effect subtracts the D_{21} portion of the columns from the right-hand side. Finally we compute $\hat{\tau} = \tau + Q_{11}^H z_1$. This simple operation merges the previous pending subtraction at the right-hand side and the subtraction of the first $m_1 - k_1$ columns (those corresponding to z_1) from the new right-hand side.

At this stage, we discard the first $m_1 - k_1$ equations and are left with a new linear system of equations

$$\hat{A}\hat{x} = \hat{b} - \begin{pmatrix} 0 \\ P_2 \\ P_3 R_2 \\ P_4 R_3 R_2 \\ \vdots \\ P_n R_{n-1} \cdots R_2 \end{pmatrix} \hat{\tau}$$

with exactly the same form as (1). To see this, we note that among the seven sequences $\{U_i\}_{i=1}^{n-1}$, $\{V_i\}_{i=2}^{n}$, $\{W_i\}_{i=2}^{n-1}$, $\{P_i\}_{i=2}^{n}$, $\{Q_i\}_{i=1}^{n-1}$, $\{R_i\}_{i=2}^{n-1}$ and $\{D_i\}_{i=1}^{n}$, everything remains the same except that U_1, Q_1, and D_1 have been replaced by $\hat{U}_1$, $\hat{Q}_1$, and D_{22}. Among the partitioned unknown subvectors x_i's and right hand side subvectors b_i's, the only changes are that x_1 and b_1 have been replaced by $\hat{x}_1$ and $\hat{b}_1$, respectively. Of course, the new linear system of equations has a strictly smaller dimension, hence we can indeed proceed with this recursion. After we have computed the unknowns x_2 to x_n and the transformed unknowns $\hat{x}_1$, we can recover x_1 using the formula

$$x_1 = w_1^H \begin{pmatrix} z_1 \\ \hat{x}_1 \end{pmatrix}.$$

Case of $k_1 \geq m_1$: Merge. We perform merging in this case. In the case $n > 1$ and $m_1 \leq k_1$, we cannot perform eliminations. Instead we merge the first two block rows and columns of A while still maintaining the sequentially semi-separable structure.

We merge the first two blocks by computing

$$\hat{D}_1 = \begin{pmatrix} D_1 & U_1 V_2^H \\ P_2 Q_1^H & D_2 \end{pmatrix}, \quad \hat{U}_1 = \begin{pmatrix} U_1 W_2 \\ U_2 \end{pmatrix}, \quad \text{and} \quad \hat{Q}_1 = \begin{pmatrix} Q_1 R_2^H \\ Q_2 \end{pmatrix}.$$

We merge x_1 and x_2 into $\hat{x}_1$, and we merge the right hand sides by computing

$$\hat{b}_1 = \begin{pmatrix} b_1 \\ b_2 - P_2 \tau \end{pmatrix} \quad \text{and} \quad \hat{\tau} = R_2 \tau.$$

Let $\hat{A}$ and $\hat{b}$ denote the matrix A and the vector b after this merge. We can rewrite (1) equivalently as

$$\hat{A}\hat{x} = \hat{b} - \begin{pmatrix} 0 \\ P_2 \\ P_3 R_2 \\ P_4 R_3 R_2 \\ \vdots \\ P_{n-1} R_{n-2} \cdots R_2 \end{pmatrix} \hat{\tau}.$$

Clearly $\hat{A}$ is again a sequentially semi-separable matrix associated with the seven hatted sequences except that we have reduced the number of blocks from n to $n-1$.

To complete the recursion, we observe that if $n = 1$, the equations (1) become the standard linear system of equations and can therefore be solved by standard solution techniques.

2.1 Flop Count

The total flop count for this algorithm can be estimated as follows. For simplicity we assume that compression and merging steps always alternate. We also assume without loss of generality that b has only one column. Then we can show that the leading terms of the flop count are given by

$$2\sum_{i=1}^{n} (m_i + k_{i-1})k_i^2 + (m_i + k_{i-1})^3 + (m_i + k_{i-1})^2 l_{i+1} \\ + k_i^2 m_{i+1} + k_i l_{i+1}(m_{i+1} + l_i + l_{i+2}).$$

To get a better feel for the operation count we look at the important case when $m_i = m$, $k_i = k$ and $l_i = l$. Then the count simplifies to

$$2n\left(m^3 + m^2(3k+l) + m(3k+l) + m(3kl+5k^2) + 2k^3 + k^2 l + 2kl^2\right).$$

We observe that the count is not symmetric in k_i and l_i. Therefore sometimes it is cheaper to compute a URV^T factorization instead. This matter is also covered in [4]. When $k = l$, the count simplifies further to

$$2n(m^3 + 4m^2k + 8mk^2 + 5k^3).$$

If we make the further assumption that $m = k$ then we get the flop count $36nk^3$. Note that the constant in front of the leading term is not large.

2.2 Experimental Run-Times

We now report the run-times of this algorithm on a PowerBook G4 running at 400 MHz with 768 MB of RAM. We used the ATLAS BLAS version 3.3.14 and LAPACK version 3 libraries. For comparison we also report the run-times of the standard dense solvers from LAPACK and ATLAS BLAS. All timings are reported in Table 2. The columns are indexed by the actual size of the matrix, which range from 256 to 8192. The horizontal rows are indexed by the value of m_i which is set equal to k_i and l_i for all i and ranges from 16 to 128. These are representative for many classes of problems (see [9]). In the last row we report the run-times in seconds of a standard dense (Gaussian elimination) solver from the LAPACK version 3 library running on top of the ATLAS BLAS version 3.3.14. These are highly-tuned routines which essentially run at peak flop rates.

From the table we can see the expected linear dependence on the size of the matrix. The non-quadratic dependence on m_i (and k_i and l_i) seems to be due to the dominance of the low-order complexity terms. For example we observe a *decrease* in run-time when we increase m_i from 64 to 128 for a matrix of size 256! This is because at this size and rank the matrix has no structure and essentially a dense solver (without any of the overhead associated with a fast solver) is being used. There is also a non-linear increase in the run-time when we increase the size from 256 to 512 for $m_i = k_i = l_i = 128$. This is due to the lower over-heads associated with standard solver.

Restricting our attention to the last two rows in Table 2 where $m_i = k_i = l_i = 128$ for all i, we observe that the fast algorithm breaks even with the dense solver for matrices of size between 512 and 1024. (The estimated flop count actually predicts a break-even around matrices of size 940.) For matrices of size 4096 we have speed-ups in excess of 17.2401. Since the standard solver becomes unusually slower for matrices of size 8192 (possibly due to a shortage of RAM) we get a speed-up of 130 at this size. The speed-ups are even better for smaller values of m_i's.

We could further speed up the fast algorithm by using Gaussian elimination with partial pivoting instead of orthogonal transforms. This approach would still be completely stable as long as the dimensions of the diagonal blocks remain small.

Table 2. Run-times in seconds for both the fast stable algorithm and standard solver for random sequentially semi-separable matrices with $m_i = k_i = l_i$ for all i

	size					
$m_i = l_i = k_i$ for all i	256	512	1024	2048	4096	8192
16	0.04	0.08	0.16	0.36	0.67	1.34
32	0.08	0.19	0.42	0.83	1.66	3.44
64	0.18	0.48	1.12	2.36	4.8	9.87
128	0.15	1.01	2.73	6.09	12.91	26.9
Standard Solver (GEPP)	0.15	0.72	4.57	30.46	222.57	3499.46

3 Constructing Sequentially Semi-separable Matrices

In this section we consider the problem of computing the sequentially semi-separable structure of a matrix given the sequence $\{m_i\}_{i=1}^n$ and a low-rank representation of some off-diagonal blocks. The second assumption is to allow for the efficient computation of the sequentially semi-separable representation of matrices possessing some other structure. The method presented can be applied to any unstructured matrix, thus proving that any matrix has a sequentially semi-separable structure (of course, k_i and l_i will usually be large in this case, precluding any speed-ups).

3.1 General Construction Algorithm

Let A represent the matrix for which we wish to construct a sequentially semi-separable representation corresponding to the sequence $\{m_i\}_{i=1}^n$, where $\sum m_i = N$, the order of the matrix. Our procedure is similar to that of Dewilde and van der Veen [12]. Since the upper triangular part and lower triangular parts are so similar, we will only describe how to construct the sequentially semi-separable representation of the strictly block upper triangular part of A. The basic idea is to recursively compress off-diagonal blocks into low-rank representations.

Let H_i denote the off-diagonal block

$$H_i = \begin{pmatrix} U_1 W_2 \cdots W_i V_{i+1}^H & \cdots & U_1 W_2 \cdots W_{n-1} V_n^H \\ \vdots & \vdots & \vdots \\ U_i V_{i+1}^H & \cdots & U_i W_{i+1} \cdots W_{n-1} V_n^H \end{pmatrix}, \tag{3}$$

and let $H_i \approx E_i \Sigma_i F_i^H$ denote a low-rank (also called economy) SVD of H_i. That is, we assume that the matrix of singular values Σ_i, is a square invertible matrix, all of whose singular values below a certain threshold have been set to zero. Therefore, E_i and F_i have an orthonormal set of columns, but they may not be unitary. Following Dewilde and van der Veen [12] we will call H_i the ith Hankel block. Each H_i is a $\mu_i \times \nu_i$ matrix with $\mu_i = m_1 + \cdots + m_i$ and $\nu_i = m_{i+1} + \cdots + m_n$.

Observe that we can obtain H_{i+1} from H_i by dropping the first m_{i+1} columns of H_i and then appending to the resulting matrix the last m_{i+1} rows of H_{i+1}. We will discuss the details of computing the SVD of H_{i+1} from that of H_i shortly.

For now, we want to compute the representation of H_i in (3) using the SVDs. Partition the SVD of $H_i \approx E_i \Sigma_i F_i^H$ as:

$$E_i = \begin{array}{c} \mu_{i-1} \\ m_i \end{array} \begin{pmatrix} E_{i,1} \\ E_{i,2} \end{pmatrix} \quad \text{and} \quad F_i = \begin{array}{c} m_{i+1} \\ \nu_{i+1} \end{array} \begin{pmatrix} F_{i,1} \\ F_{i,2} \end{pmatrix}. \tag{4}$$

Observe that we can pick $U_i = E_{i,2}$ and $V_{i+1} = F_{i,1} \Sigma_i^H$ (of course $\Sigma_i^H = \Sigma_i$).

How do we pick W_i? Observe that H_{i+1} and H_i share a large block of the matrix. It follows from the sequentially semi-separable representation that we should pick W_{i+1} such that $E_i W_{i+1}$ will give a column basis for the upper portion of H_{i+1}. It follows that we should pick W_{i+1} to satisfy the following requirement, $E_i W_{i+1} = E_{i+1,1}$. We can solve this easily to obtain $W_{i+1} = E_i^H E_{i+1,1}$.

The proof that these formulas work can be seen by substituting them back into the sequentially semi-separable representation beginning with H_1.

However, we are still not done. To compute the sequentially semi-separable representation efficiently it is important to compute the SVD of H_{i+1} quickly. To do that we need to use the SVD of H_i. As we mentioned earlier, H_{i+1} is obtained from H_i by dropping the first m_{i+1} columns of H_i and then appending to the resulting matrix the last m_{i+1} rows of H_{i+1}, which we will call Z. Hence we can rewrite H_{i+1} in the notation of (4) as

$$H_{i+1} \approx \begin{pmatrix} E_i \Sigma_i F_{i,2} \\ Z \end{pmatrix} = \begin{pmatrix} E_i & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \Sigma_i F_{i,2} \\ Z \end{pmatrix}.$$

Hence we compute the low-rank SVD $\begin{pmatrix} \Sigma_i F_{i,2} \\ Z \end{pmatrix} \approx \tilde{E}\tilde{S}\tilde{F}^H$ and obtain the low-rank SVD of H_{i+1} as follows:

$$H_{i+1} \approx \left(\begin{pmatrix} E_i & 0 \\ 0 & I \end{pmatrix} \tilde{E} \right) \tilde{S}\tilde{F}^H.$$

Finally, we note that the sequentially semi-separable representation for the lower triangular part of A can be computed by applying exactly the same procedure above to A^H. The computational costs are similar as well.

This algorithm takes $O(N^2)$ flops, where the hidden constants depend on m_i, k_i and l_i. The algorithm can be implemented to require only $O(N)$ memory locations. This is particularly important in those applications where a large dense structured matrix can be generated (or read from a file) on the fly. Many computational electromagnetics problems involving integral equations fall in this class.

We can replace the use of singular value decompositions with rank-revealing QR factorizations (QR factorizations with column pivoting) quite easily. This may result in some speed ups with little loss of compression. The only difficulty might be the lack of easily available software.

A totally different alternative is to use the recursively semi-separable (RSS) representation presented in the paper by Chandrasekaran and Gu [4]. This is usually easier to compute efficiently, but may be less flexible.

In many important applications the sequentially semi-separable representation needs to be computed only once for a fixed problem size and stored in a file. In such cases the cost of the exact algorithm is not important. Such cases include computing the sequentially semi-separable structure of spectral discretization methods of Greengard and Rokhlin [23, 29] for two-point boundary value problems and that of Kress [11] for integral equations of classical potential theory in two dimensions.

4 Two-Dimensional Scattering

For two-dimensional exterior scattering problems on analytic curves for acoustic and electro-magnetic waves, Kress' method of discretization of order $2n$ will lead to a $2n \times 2n$ matrix of the form

$$A \ = I + R \odot K_1 + K_2,$$

where K_1 and K_2 are low-rank matrices and

$$R_{ij} = -\frac{2\pi}{n} \sum_{m=1}^{n-1} \frac{1}{m} \cos \frac{m|i-j|\pi}{n} - \frac{(-1)^{|i-j|}\pi}{n^2}.$$

From the results in [9] we see that it is sufficient to verify that R is a sequentially semi-separable matrix of low Hankel-block ranks. It would then follow that A is a sequentially semi-separable matrix of low Hankel-block ranks. In Table 3 we exhibit the peak Hankel block ranks of R. The rows are indexed by the (absolute) tolerance we used to determine the numerical ranks of the Hankel blocks. In particular we used tolerances of 10^{-8} and 10^{-12} that are useful in practice. The columns are indexed by the size of R.

As can be seen the ranks seem to depend logarithmically on the size N, of R. This implies that the fast algorithm will take $O(N \log^2 N)$ flops to solve linear systems involving A. We observe that the sequentially semi-separable representations of R for different sizes and tolerances need to be computed once and stored off-line. Then using the results in [9] we can compute the sequentially semi-separable representation of A rapidly on the fly.

Acknowledgements

The authors are grateful to Robert Sharpe and Daniel White of Lawrence Livermore National Laboratory for their suggestions and interests in this research.

Table 3. Peak Hankel block ranks for the spectral method of Kress, Martensen and Kussmaul for the exterior Helmholtz problem.

	size					
tolerance	256	512	1024	2048	4096	8192
1E-8	28	32	34	37	38	40
1E-12	40	46	52	58	62	66

References

[1] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff, *Algorithm 679: A Set of Level 3 Basic Linear Algebra Subprograms: Model Implementation and Test Programs*, ACM Transactions on Mathematical Software, 16 (1990), pp. 18-28.

[2] F. X. Canning and K. Rogovin, *Fast direct solution of moment-method matrices*, IEEE Antennas and Propagation Magazine, 40 (1998).

[3] J. Carrier, L. Greengard, and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 669–686.

[4] S. Chandrasekaran and M. Gu, *Fast and stable algorithms for banded plus semi-separable matrices*, submitted to SIAM J. Matrix Anal. Appl., 2000. 545, 546, 549, 552

[5] S. Chandrasekaran and M. Gu, *A fast and stable solver for recursively semi-separable systems of equations*, in *Structured matrices in mathematics, computer science and engineering, II*, edited by Vadim Olshevsky, in the Contemporary Mathematics series, AMS publications, 2001.

[6] S. Chandrasekaran and M. Gu, *Fast and Stable Eigendecomposition of Symmetric Banded plus Semi-separable Matrices*, 1999, *Linear Algebra and its Applications*, Volume 313, Issues 1-3, 1 July 2000, pages 107-114.

[7] S. Chandrasekaran and M. Gu, *A Divide-and-Conquer Algorithm for the Eigendecomposition of Symmetric Block-Diagonal Plus Semiseparable Matrices*, 1999, accepted for publication in *Numerische Mathematik*.

[8] S. Chandrasekaran, M. Gu, and T. Pals, *A fast and stable solver for smooth recursively semi-separable systems*. Paper presented at the SIAM Annual Conference, San Diego, CA, 2001, and SIAM Conference of Linear Algebra in Controls, Signals and Systems, Boston, MA, 2001.

[9] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A.-J. van der Veen, *Fast and stable solvers for sequentially recursively semi-separable linear systems of equations*. Submitted to SIAM Journal on Matrix Analysis and its Applications, 2002. 549, 552

[10] D. Colton and R. Kress, *Integral Equation Methods in Scattering Theory*, Wiley, 1983.

[11] D. Colton and R. Kress, *Inverse acoustic and electromagnetic scattering theory,* Applied Mathematical Sciences, vol. 93, Springer-Verlag, 1992. 545, 552

[12] P. Dewilde and A. van der Veen, *Time-varying systems and computations*. Kluwer Academic Publishers, 1998. 545, 546, 550

[13] Y. Eidelman and I. Gohberg, *Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices*, Computers and Mathematics with Applications, 33 (1997), Elsevier, pp. 69–79.

[14] Y. Eidelman and I. Gohberg, *A look-ahead block Schur algorithm for diagonal plus semiseparable matrices*, Computers and Mathematics with Applications, 35 1998), pp. 25–34.
[15] Y. Eidelman and I. Gohberg, *A modification of the Dewilde van der Veen method for inversion of finite structured matrices,* Linear Algebra and its Applications, volumes 343–344, 1 march 2001, pages 419–450.
[16] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comp. Phys., 73 (1987), pp. 325–348.
[17] L. Gurel and W. C. Chew, *Fast direct (non-iterative) solvers for integral-equation formulations of scattering problems,* IEEE Antennas and Propagation Society International Symposium, 1998 Digest, Antennas: Gateways to the Global Network, pp. 298–301, vol. 1.
[18] W. Hackbusch, *A sparse arithmetic based on $\mathcal{H}$-matrices. Part-I: Introduction to $\mathcal{H}$-matrices*, Computing 62, pp. 89–108, 1999.
[19] W. Hackbusch and B. N. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic. Part-II: application to multi-dimensional problems*, Computing 64, pp. 21–47, 2000.
[20] W. Hackbusch and B. N. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic: general complexity estimates,* Journal of Computational and Applied Mathematics, volume 125, pp. 79–501, 2000.
[21] D. Gope and V. Jandhyala, *An iteration-free fast multilevel solver for dense method of moment systems,* Electrical Performance of Electronic Packaging, 2001, pp. 177–180.
[22] R. Kussmaul, *Ein numerisches Verfahren zur Lösung des Neumannschen Aussenraumproblems für die Helmholtzsche Schwingungsgleichung,* Computing 4, 246–273, 1969.
[23] June-Yub Lee and L. Greengard, *A fast adaptive numerical method for stiff two-point boundary value problems,*SIAM Journal on Scientific Computing, vol.18, (no.2), SIAM, March 1997, pp. 403–29. 552
[24] E. Martensen, *Über eine Methode zum räumlichen Neumannschen Problem mit einer Anwendung für torusartige Berandungen,* Acta Math. 109, 75–135, 1963.
[25] N. Mastronardi, S. Chandrasekaran and S. van Huffel, *Fast and stable two–way chasing algorithm for diagonal plus semi-separable systems of linear equations*, *Numerical Linear Algebra with Applications*, Volume 38, Issue 1, January 2000, pages 7–12.
[26] N. Mastronardi, S. Chandrasekaran and S. van Huffel, *Fast and stable algorithms for reducing diagonal plus semi-separable matrices to tridiagonal and bidiagonal form*, *BIT*, volume 41, number 1, March 2001, pages 149–157.
[27] T. Pals. Ph. D. Thesis, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, 2002.
[28] V. Rokhlin, *Applications of volume integrals to the solution of PDEs*, J. Comp. Phys., 86 (1990), pp. 414–439.
[29] P. Starr, *On the numerical solution of one-dimensional integral and differential equations*, Thesis advisor: V. Rokhlin, Research Report YALEU/DCS/RR-888, Department of Computer Science, Yale University, New Haven, CT, December 1991. 552