INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

NEWTON'S ITERATION FOR STRUCTURED MATRICES

by

RHYS ERIC ROSHOLT

A dissertation
submitted to the Graduate Faculty in Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
The City University of New York

2003

This manuscript
has been read and accepted for the Graduate Faculty in Computer Science
in satisfaction of the dissertation requirement for the degree of
Doctor of Philosophy.

4/22/03 — Date

Victor Pan — Chair of Examining Committee

4/23/03 — Date

Executive Officer

Dr. Victor Pan, Adviser and Chair
The Graduate School and University Center, Lehman College

Dr. Theodore Brown
The Graduate School and University Center, Queens College

Dr. Stefan Burr
The Graduate School and University Center, The City College

Dr. Stathis Zachos
The Graduate School and University Center, Brooklyn College

Dr. Ailong Zheng
Bell Labs, Lucent Technologies

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

NEWTON'S ITERATION FOR STRUCTURED MATRICES

by

RHYS ERIC ROSHOLT

Adviser: Dr. Victor Pan

In the first part of the thesis (Chapters 1-5), we study four quadratically convergent algorithms for the refinement of rough initial approximations to the inverses of $n \times n$ nonsingular Toeplitz and Toeplitz-like matrices and to the solutions of nonsingular Toeplitz and Toeplitz-like linear systems of n equations. The algorithms are variations of Newton's iteration, adjusted to structured matrix computations based on the known inversion formulas for Toeplitz matrices and the displacement representations of Toeplitz-like matrices. Due to using matrix structure, each iteration step is performed in $O(n \log n)$ time. The algorithms can be extended to the case of structured matrices of other classes.

In the second part of the thesis (Chapters 6-11), we study more general residual correction algorithms for the computation of the inverses of general and structured matrices. For structured matrices, we extend the algorithms by new policies of compression delay. For both structured and unstructured matrices, we study the homotopic (continuation) methods, which supply close initial approximations. For unstructured indefinite complex Hermitian input matrices, the homotopy methods enable substantial acceleration of the known best non-homotopic algorithms. Furthermore, by using homotopic techniques, we guarantee superlinear convergence to the inverses of structured matrices even where no initial approximation is available and where compression of displacement generators is ensured in every residual correction step.

Numerical experiments on the initial four algorithms confirm the results of our analysis and the efficiency of the algorithms. Numerical tests with Toeplitz input matrices show a greater power of both homotopic and non-homotopic approaches than the theoretical study predicts.

Acknowledgements

I would like to thank the following individuals:

My adviser, Dr. Victor Pan, for years of hard work teaching me how to do research, write papers, review proposed papers, write grant applications, review grant applications, and stay focused. His generousity, friendliness, accessibility, and thorough-going knowledge of this material has been an inspiration and an example.

Dr. Theodore Brown, the Executive Officer of the Doctoral Program in Computer Science, for his help and understanding. His knowledge of algorithms and his teaching of analysis of algorithms was important to my doctoral studies. His leadership has resulted in continued advancement and growth of both the program and the CUNY Institute for Software Design and Development.

Dr. Stanley Habib, who was the Executive Officer of the Doctoral Program in Computer Science when I was admitted to the program. His trust and faith in me led him to appoint me as student representative to the Executive Committee of the program and to the Graduate Council, where I gained experience that very few students receive.

Dr. Connor Lazarov, who was the graduate adviser for the Department of Mathematics and Computer Science, Lehman College, The City University of New York, when I took my Master's Degree program there. He was instrumental in my decision to pursue Computer Science. He supported my application to join the faculty of Lehman College, and he helped me to fit in and feel welcome in that lofty environment.

Dr. Robert Feinerman, Chairman of the Department of Mathematics and Computer Science, Lehman College, The City University of New York. He is the most supporting leader of a department that I can imagine.

I would also like to thank the faculty of the Doctoral Program in Computer Science at The Graduate School and University Center of The City University of New York, and the faculty of the Department of Mathematics and Computer Science at Lehman College of The City University of New York.

Finally, I would like to thank my parents, Karlton Jerome Rosholt and Carolyn Janell (Hanson) Rosholt. The love, support, encouragement and forgiveness they showed me have served me well.

Neither Dr. Connor Lazarov nor my mother lived to see this. I miss them.

Contents

List of Tables

1 Introduction

Preconditioned conjugate gradient (PCG) methods are quite effective for the solution of a large class of Toeplitz and Toeplitz-like linear systems of equations. In a few iterations they provide an approximate solution and then improve the approximations with linear convergence rate. While this performance is sufficient in several applications, there are many cases where it is still desirable to compute a highly accurate solution at a faster convergence rate, assuming that a rough initial approximation has been computed by a PCG algorithm. Alternatively, any Toeplitz and Toeplitz-like linear system of equations can be effectively solved by the known direct methods [AGr88], [Bun], [GKO], [G] (the two latter papers extending the general approach of [P90] to yield practical Toeplitz and Toeplitz-like solvers), but in many cases the computed solution must be refined to counter the effect of rounding errors.

We rely on the techniques of [P90], [P92], [P93], [P93a], [PZHD97], and on their variations to compute the desired refinement by means of Newton's iteration, which represents the class of the residual correction methods and which we simplify in the case of Toeplitz and Toeplitz-like matrices. Our first two modifications of Newton's iteration (see Algorithms 1 and 2) exploit the displacement structure of the input matrices to simplify the computations. They work in the more general case of a Toeplitz-like input. Our third and fourth modifications (see Algorithms 3 and 4) specialize Newton's iteration as the residual correction of the solution of a Toeplitz linear system of equations. They rely on the inversion formulae known for Toeplitz matrices and simplify Algorithms 1 and 2, respectively.

Algorithm 1 is a little more costly to perform but implies the convergence under milder assumptions about the initial approximation than Algorithm 2. Algorithm 4 runs roughly twice as fast as Algorithm 3 and does not seem to require

any stronger initial assumptions. Algorithms 1 and 3 are more convenient to apply where the triangular Toeplitz representation is used for the input and output matrices, whereas Algorithms 2 and 4 are more convenient to apply where a factor-circulant (f-circulant) representation is used. As we will point out in the concluding remarks, at least some of our algorithms can be extended to other classes of structured matrices.

All presented algorithms require an initial approximation lying sufficiently close to the solution (see equations (5.5), (5.9), (6.9)). Otherwise the iterations may converge too slowly or diverge. A partial remedy can be obtained by means of homotopy techniques (cf. our Remark 5.1 in Section 5.3 and [P92]).

1.1 Newton's Iteration for Matrix Inversion

Newton's iteration is a major tool for solving a functional equation, $f(x) = 0$. Let x_0 be an initial approximation to a solution x. Then the iteration successively produces a sequence of iterates $x_{i+1} = x_i - f(x_i)/f'(x_i)$, with errors $e_i = |x_i - x|$, $i = 0, 1, \ldots = s$. If $f(x)$ is smooth enough in a neighborhood of x and if e_0 is small enough, then the approximations converge to the solution with quadratic convergence rate: $e_{i+1} = O(e_i^2)$, $i = 0, 1, \ldots$ (see e.g. [A]).

For example, suppose that we wish to apply Newton's iteration to approximate $1/t$ for a given positive binary value $t = 2^e(0.t_1t_2t_3\ldots)_2$, where e is an integer, $t_1 = 1$, $t_i \in \{0, 1\}$ for $i = 2, 3, \ldots$. By choosing $x_0 = 2^{-e}$ and $f(x) = t - 1/x$, we obtain $0 < |1 - tx_0| = 1 - (0.t_1t_2\ldots)_2 \leq 1/2$, and Newton's iteration takes the form of the recursive computation of the values $x_i = x_{i-1} - (t - 1/x_{i-1})/(1/x_{i-1}^2) = x_{i-1}(2 - tx_{i-1})$, $i = 1, 2, \ldots$. It follows that $1 - tx_i = 1 - 2tx_{i-1} + (tx_{i-1})^2 = (1 - tx_{i-1})^2$. Recursively, we obtain that $0 < 1 - tx_i = (1 - tx_0)^{2^i} \leq 2^{-2^i}$, $i = 1, 2, \ldots$. Therefore, $0 < 1/t - x_i \leq (1/t)2^{-2^i}$, that is, the iteration very rapidly (quadratically) converges to $1/t$.

Table 1.1: Newton's iteration for the reciprocal of 5/8.

i	x_i	$1-tx_i$	$1/t-x_i$
0	1.0000000e+00	3.7500000e-01	6.0000000e-01
1	1.3750000e+00	1.4062500e-01	2.2500000e-01
2	1.5683594e+00	1.9775391e-02	3.1640649e-02
3	1.5993743e+00	3.9106607e-04	6.2572956e-04
4	1.5999998e+00	1.3411045e-07	2.3841858e-07

For instance, let $t = (0.101)_2 = 1/2 + 1/8$. Then the results of performing four steps of Newton's iteration on a computer with the 8-decimal precision can be summarized in the next table (where the values $1-tx_4$ and $1/t-x_4$ are shown without several decimal digits lost at the stage of the subtraction of $tx_4 = 0.99999999$ from 1).

Remark 1.1. (cf. [B68]). *This iteration can be traced back four millennia, when the ancient Babylonians applied it in rudimentary form to solving quadratic equations. For the solution of polynomial equations of higher degree, the iteration was routinely used by Chinese and Arab mathematicians in medieval time. Among Europeans before Newton, Francoise Viete should be mentioned. The current version of the iteration was given by Joseph Raphson in 1690. We use Newton's name to be consistent with the commonly accepted (though inaccurate) terminology.*

Newton's iteration can be also effectively applied to some equations $f(X) = 0$ where X is a matrix rather than a scalar. We will next show the application to the inversion of an $n \times n$ non-singular matrix A. The history of such an application of Newton's iteration can be traced back to [s33]. A more recent treatment can be found in [ps91]. We will assume that A is given with an initial approximation

$-X_0$ to its inverse A^{-1} satisfying the bound

$$\tau = ||I + X_0 A|| = ||I - (-X_0)A|| < 1 ,$$

for some positive τ and some fixed matrix norm. An initial approximation $-X_0$ to A^{-1} is assumed to be supplied by another computational process, for example, by a PCG algorithm, by an exact solution algorithm [GKO], [G], which requires refinement due to rounding errors, or by the homotopy algorithm of [P92]. In this case, $f(X) = A + X^{-1}$, and the iteration takes the form of the recursive computation of the matrices

$$X_{i+1} = X_i(2I + AX_i) = (2I + X_i A)X_i , \ i = 0, 1, \ldots . \tag{1.1}$$

We deviate slightly from the customary format of Newton's iteration by writing $-X_i$ instead of X_i, to denote the computed approximations to the inverse matrix A^{-1}.

It follows from (1.1) that

$$I + X_{i+1}A = (I + X_i A)^2 = (I + X_{i-1}A)^4 = \cdots = (I + X_0 A)^{2^{i+1}} .$$

Consequently,

$$||I + X_i A|| = ||I - (-X_i)A|| \leq \tau^{2^i} ,$$

so that the norm of the residual matrix $I + X_i A$ is bounded from above by τ^{2^i}. This establishes quadratic convergence of the iteration. Given a desired residual norm bound $\epsilon > 0$, the bound

$$||I + X_i A|| < \epsilon \tag{1.2}$$

is guaranteed in $\lceil \log_2(\frac{\log \epsilon}{\log \tau}) \rceil$ recursive steps (1.1), where $\lceil x \rceil$ stands for the smallest integer not exceeded by x. We also deduce from (1.2) that $||A|| \cdot ||A^{-1} + X_i|| \leq ||I + X_i A|| < \epsilon$, and since $||A^{-1}|| \cdot ||A|| \geq 1$, it follows that

$$\frac{||A^{-1} - (-X_i)||}{||A^{-1}||} < \epsilon ,$$

so that the matrix $-X_i$ approximates A^{-1} with a relative error norm less than ϵ.

Each step of the iteration (1.1) involves the multiplication of A by X_i, the addition of 2 to all the diagonal entries of the product (which takes n additions), and the pre-multiplication of the resulting matrix by X_i. The most costly steps are the two matrix multiplications, each taking n^3 multiplications and $n^3 - n^2$ additions for a pair of general $n \times n$ input matrices A and X_0.

The next example demonstrates the advantage of the quadratic convergence rate over the linear convergence rate.

Example. Let $\tau = 1/2$, $\epsilon = 10^{-6}$. Then 5 iteration steps (1.1) suffice to compute the matrix X_5 satisfying the bounds $||I + X_5 A|| < \epsilon$ and $||A^{-1} - (-X_5)|| < \epsilon ||A^{-1}||$. On the other hand, assume any linearly convergent scheme such as Jacobi's or Gauss-Seidel's iteration (cf. [GL96, p. 510], where for the same input the norm of the error matrix of the computed approximation to A^{-1} decreases by factor 2 in each iteration step. Then the error norm decreases below 2^{-i-1} in i iteration steps, so it takes 19 steps to ensure the bound (1.2) for $\varepsilon = 10^{-6}$.

1.2 Residual Correction (RC) Processes

We study *residual correction processes* converging to the inverse or the Moore–Penrose generalized inverse of a general $n \times n$ matrix M [S33], [B-I66], [B-IC66], [IK66], [SS74], [PS91]. The basic processes perform matrix multiplication p times in each step to achieve convergence of the order of p, for any $p \geq 2$. With appropriate scaling, however, one may reach the order of $p > 2$ by performing matrix multiplication only twice per step [PS91]. Hereafter, we write RC for "residual correction" and MM for "matrix multiplication". For $p = 2$ unscaled RC processes turn into *Newton's iteration*. The RC processes can be directed to converge to numerical generalized inverses and are known for their strong numerical stability and self-correcting property [PS91].

For simplicity (until Section 11) we assume non-singularity of the input matrices M. Here are the two main problems with RC processes.

a) The RC processes require additional techniques for the computation of an initial approximation to the inverse. The known techniques of [B-I66], [B-IC66], [SS74], and [PS91] produce a crude initial approximation. Then it takes quite a few, the order of $\log_2 \kappa(M)$, RC steps ($\kappa(M) = \text{cond}(M)$ denoting the condition number of the matrix M) to refine the approximation to the level from which the iteration very rapidly converges. Our main topic is an alternative approach based on the homotopic (continuation) techniques.

b) For general matrices, MM is an expensive operation, comparable to matrix inversion in its computational cost, although substantially simpler than the computation of the generalized inverse and allowing effective parallel implementation.

However, MM is dramatically simplified in the highly important case of structured matrices, represented by their displacements in a compressed form. Namely, the displacement of an $n \times n$ matrix occupies memory space $O(n)$, and multiplication of $n \times n$ compressed structured matrices uses $O(n \log n)$ or $O(n \log^2 n)$ flops. Consequently, the RC processes can be also performed by using small memory space and little computer time as long as matrix structure and compressed representation of matrices are preserved throughout the computation without destroying rapid convergence of the process. Here some advanced compression techniques are applied, first proposed in [P92] and then elaborated upon in [PZHD97], [PBRZ99], [PR01], [PRW01], [P01a].

We propose some new techniques to improve practical performance of the known algorithms in the case of structured input matrices. These techniques rely

on *delayed compression* and *modular arithmetic in the real field* (see Section 2.4, Remark 3.2, and equations (3.8)–(3.11)).

1.3 Homotopic RC Processes

The solution techniques for problems a) and b) do not always match one another. That is, compression perturbs computed approximations and may easily destroy convergence at the initial stages of the RC processes where the convergence is fragile. This implies some additional requirements. We must either yield much closer initial approximations versus the known techniques of [B-I66], [B-IC66], [SS74], and [PS91] or perform much more work per iteration step to yield compression with much smaller perturbation of the computed approximations to the inverse. The original approach in [P92] achieves the latter goal for the class of Toeplitz and Toeplitz-like matrices by developing the *homotopic* (or *continuation*) method but also allows a very natural heuristic modification towards this goal. The approach in [P92] relies on truncating the smallest singular values of the displacements. Tests show that the heuristic is surprisingly effective in the important case of Toeplitz input matrices, but no theoretical results support this development.

We recall the RC algorithms for general and structured matrices, show their improved variations for Toeplitz and other sturctured matrices (see (3.8))–(3.11)), and report the results of numerical experiments in Appendix B; otherwise our main subjects are new techniques for computing the initial approximation. Their efficiency is confirmed by both experiments and the proved estimates for the computational work. The new methods extend the homotopic (continuation) techniques of [P92] relying on the inversion of a readily invertible matrix M_0 (e.g., $M_0 = I$) and the subsequent homotopic transition to the matrix M along the trajectories

$$M_h = (1 - t_h)M + t_h M_0, \ h = 0, 1, \ldots, \tag{1.3}$$

or

$$M_h = M + t_h M_0, \ h = 0, 1, \ldots, \tag{1.4}$$

where

$$t_0 > t_1 > \ldots > t_H = 0, \tag{1.5}$$

t_0 is 1 in (1.3) and a sufficiently large value in (1.4). We arrange the homotopy to keep the trajectories $M(t)$ away from singular matrices for $t_0 \geq t \geq 0$; we prove that for $t \geq 0$ the condition numbers of the matrices $M(t)$ reach their maximums where $t = 0$ (cf. our earlier *techniques of variable diagonal* [P00b]). By choosing the step sizes $t_h - t_{h+1}$ sufficiently small, we may always ensure that the matrix $M_h^{-1} M_{h+1}$ is close enough to the identity matrix. Then the approximation to the inverse M_h^{-1} computed at the h-th homotopic step would serve as a good initial approximation at the $(h+1)$-st homotopic step.

2 Residual Correction (RC) Processes

Hereafter, M^T, $\mathbf{v}^T$, M^*, and $\mathbf{v}^*$ denote the transposes and Hermitian (conjugate) transposes of a matrix M and a vector $\mathbf{v}$, respectively. We write $\sigma_j = \sigma_j(M)$, $\kappa(M) = \sigma_1/\sigma_r$. σ_j denote the singular values of a matrix M where $r = \text{rank}(M)$, $j = 1, \ldots, r$; $0 < \sigma_- \le \sigma_r \le \ldots \le \sigma_1 \le \sigma_+$; $\kappa(M)$ is the condition number of M. $\mathbf{e}_{i-1}$ denotes the i-th coordinate vector, $i = 1, \ldots, n$. $\lceil x \rceil$ is the smallest among the integers not exceeded by a real x.

2.1 A Basic RC Process

A sufficiently close initial approximation X_0 to the inverse of a non-singular matrix M can be rapidly improved by means of a scaled RC process [IK66]

$$\Delta_i = \Delta(X_i) = X_{i+1} - c_{i+1}X_i = c_{i+1}\sum_{k=1}^{p-1} R_i^k X_i, \quad i = 0, 1, \ldots, \tag{2.1}$$

where we write

$$R_i = R(M, X_i) = I - X_i M = R_{i-1} - M(X_{i+1} - X_i). \tag{2.2}$$

For $p = 2$, $c_{i+1} = 1$ for all i, we arrive at Newton's iteration [S33]. For $p = 2^h$, we may compute $\sum_{k=0}^{p-1} R_i^k$ as $\prod_{j=0}^{h-1}(I + R_i^{2^j})$ using fewer additions. Already for the unscaled process, that is, under the simplest choice of

$$c_i = 1 \text{ for all } i, \tag{2.3}$$

(2.1) and (2.2) imply that

$$R_i = (R_0)^{p^i}, \quad \|R_i\| \le \|R_0\|^{p^i}, \quad i = 1, 2, \ldots. \tag{2.4}$$

That is, the unscaled RC process (2.1), (2.3) converges with the order p to the matrix M^{-1} provided that

$$\|R_0\|_2 \le \theta < 1, \quad R_0 = R(M, X_0).$$

Suppose that the latter bound holds for a fixed θ. Then the computational work required to ensure the desired upper bound on the norm $||R_i||$ is minimized for $p = 3$ [IK66, pages 86-88].

For an initial approximation X_0 to the matrix M^{-1}, one may choose [B-I66], [SS74]

$$X_0 = c_0 M^*, \quad c_0 = \frac{2}{\sigma_+^2 + \sigma_-^2} \tag{2.5}$$

to yield that

$$||R_0||_2 \leq 1 - \frac{2}{1+\kappa_+^2}, \quad \kappa_+ = \kappa_+(M) = \sigma_+/\sigma_-. \tag{2.6}$$

Now it follows that the first

$$i = 2\log_p \kappa_+ + O(1)$$

unscaled *critical RC steps* (2.1), (2.3) decrease the residual norm $||R_i||_2$ below $1/2$, and then the

$$j = \lceil \log_p \log_2(1/\epsilon) \rceil$$

additional *refinement RC steps* (2.1), (2.3) decrease the norm below any fixed positive $\epsilon \leq 1/2$ [SS74]. In Section 7.3, we use the threshold value $1/e = 0.367819\ldots$ instead of $1/2$; this may change i at most by 1.

The asymptotic bound $i_- = \log_2 \kappa(M) + O(1)$ on the number of critical RC steps is achieved in [B-I66] under the simpler initial choice of

$$X_0 = M^*/(||M||_1 ||M||_\infty).$$

Furthermore, for a Hermitian (or real symmetric) and positive definite matrix M, one may further decrease the number of critical RC steps (2.7) roughly by twice [PS91] because we have

$$||R_0||_2 \leq 1 - \frac{1}{\sqrt{n}\kappa(M)} \quad \text{for} \quad X_0 = I/||M||_F \tag{2.7}$$

where $||M||_F = \text{trace}\,(M^*M)$ denotes the Frobenius norm of the matrix M, and M is a Hermitian and positive definite matrix.

2.2 Scaled Newton's Iteration

The choice of c_{i+1} in (2.1) was optimized in [PS91] in the case of RC process (2.1) for $p = 2$:

$$X_{i+1} = c_{i+1}(I + R_i)X_i = c_{i+1}(2X_i - X_iMX_i). \tag{2.8}$$

Namely, by choosing $p = 2$,

$$c_0^- = \frac{2\sigma_-^2}{\sigma_-^2 + \sigma_+^2}, \quad c_{i+1} = \frac{2}{1 + (2 - c_i^-)c_i^-}, \quad c_{i+1}^- = (2 - c_i^-)c_i^- c_{i+1} \tag{2.9}$$

for $i = 0, 1, \ldots$, one obtains that

$$||R_l||_2 \le \max_{\sigma_- \le x \le \sigma_+} |T_{2^l}(\gamma x + \delta)|/|T_{2^l}(\delta)| \le \frac{1}{|T_{2^l}(\delta)|} \tag{2.10}$$

where $\gamma = 2/(\sigma_+ - \sigma_-)$, $\delta = -(\sigma_+ + \sigma_-)/(\sigma_+ - \sigma_-) = -1 - \gamma\sigma_-$, and $T_j(x) = \cos(j \arccos x)$ is the j-th degree Chebyshev polynomial of the first kind on the closed real interval $[-1, 1]$. It follows [FF63, Chapter 9, Section 9] that

$$||R_l||_2 \le \frac{2}{(\delta + \sqrt{\delta^2 - 1})^L + (\delta - \sqrt{\delta^2 - 1})^L}, \quad L = 2^l.$$

This bound is substantially smaller than δ^L. The number of critical steps decreases roughly by twice versus policy (2.3), reaching the level

$$i = \log_2 \kappa_+(M) + O(1/\kappa_+^2(M)). \tag{2.11}$$

In other words, the impact of the optimal scaling of (2.9) is equivalent to increasing the order of convergence of the critical steps from $q = 2$ to $q = 4$.

2.3 RC Processes for Numerical Generalized Inverse

The paper [PS91] also proposes a modification where Newton's RC processes for $p = 2$ converge to a numerical generalized (Moore–Penrose) inverse M_ϵ^+, that is, the generalized inverse of the matrix M_ϵ formed via truncating the smallest singular

values of M (up to a fixed tolerance ϵ). This is achieved by first applying iteration (2.8)–(2.9) with

$$c_0 = \sigma_+ c,\ c_0^- = c\epsilon^2,\ c = \min(2/(\sigma_+ + \epsilon^2), \rho/\epsilon^2), \tag{2.12}$$

$\rho = (1 + \sqrt{3}/2 = 1.366\ldots$ (Under the scaling of (2.12), the value ρ partitions the range for the spectrum of the matrix $X_0 M$; the partition is induced by the respective partition by ϵ of the singular values of the matrix M. Note that the bound σ_- is not needed in this variation of the iteration.) The iteration is performed until we arrive at $c_i^- \geq \rho$ for some integer i. Then the matrix X_i is scaled, that is, replaced by the matrix $(\rho/c_i^-)X_i$, and the iteration is continued based on the expressions

$$X_{i+1} = (-2X_i M + 3I)X_i M X_i,\quad i = 0, 1, \ldots. \tag{2.13}$$

(This is a generalizations of RC process (2.1) and a special case of a more general process

$$X_{i+1} = p_i(X_i M)X_i \tag{2.14}$$

where $p_i(y)$ are selected polynomials, $i = 0, 1, \ldots.$) Based on (2.13), the singular values $\sigma_j(M)$ are partitioned by ϵ into two groups: those exceeding ϵ correspond to the eigenvalues $\lambda^{(i)}$ of $X_i M$ that lie in the interval $1/2 < \lambda^{(i)} \leq \rho$; iteration (2.13) sends them towards 1. The other eigenvalues of $X_i M$ lie in the interval $[0, 1/2)$; they correspond to the singular values $\sigma_j(M) < \epsilon$. Iteration (2.12) sends them towards 0. This is exactly the desired convergence to the matrix M_ϵ^+. Convergence is ultimately quadratic but is slow near $1/2$ and ρ. Iteration can be immediately extended to computing the matrices $M_\epsilon = MM_\epsilon^+ M$ and $\tilde{M}_\epsilon = M - M_\epsilon$ and the numerical rank trace$(M_\epsilon M_\epsilon^+)$.

2.4 Bounding The Precision of Computing

It was proved in [PS91] that both original and modified Newton's processes are numerically stable. Process (2.1), however, involves the expression

$$c_{i+1}(I + \sum_{k=0}^{p-1} R_i^k)X_i,$$

whose representation for a smaller $\|R_i\|$ requires the p-fold precision versus the single precision for representation of M and X_i. For $p = 2$, $c_{i+1} = 1$, the precision growth in the RC process (2.1) can be avoided based on using *modular arithmetic in the real field* [P92b], [EPY98].

For the task of solving a linear system $M\mathbf{x} = \mathbf{b}$:

$$\mathbf{x}_1 = X_0\mathbf{b}, \quad \mathbf{r}_1 = \mathbf{b} - M\mathbf{x}_1, \tag{2.15}$$

the precision can be controlled if we apply iterative improvement process

$$\Delta_i = \mathbf{x}_{i+1} - \mathbf{x}_i = X_0\mathbf{r}_i, \quad \mathbf{r}_{i+1} = \mathbf{r}_i - M(\mathbf{x}_{i+1} - \mathbf{x}_i), \quad i = 1, \ldots, s. \tag{2.16}$$

This process invloves neither residual matrices R_i nor higher precision approximations X_i to M^{-1}, but the approximation error norm $\|\mathbf{x} - \mathbf{x}_{i-1}\|$ decreases by the factor of $\|R_0\| = \|I - X_0M\|$ in each iteration step: $\mathbf{x} - \mathbf{x}_i = (I - X_0M)(\mathbf{x} - \mathbf{x}_{i-1}) = (I - X_0M)^i(\mathbf{x} - \mathbf{x}_0)$. That is, the process converges linearly. The computations can be performed with a single/double precision, where the output vector $\mathbf{x}_s = \mathbf{x}_1 + \sum_{i=1}^{s-1} \Delta_i$ is represented by the sequence $\mathbf{x}, \Delta_1, \ldots, \Delta_{s-1}$. This is an advantage of process (2.15)–(2.16) versus processes (2.1).

Table 3.1: Four classes of structured matrices

Toeplitz matrices $(t_{i-j})_{i,j=0}^{n-1}$	Hankel matrices $(h_{i+j})_{i,j=0}^{n-1}$
$\begin{pmatrix} t_0 & t_{-1} & \cdots & t_{1-n} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{pmatrix}$	$\begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \iddots & h_n \\ \vdots & \iddots & \iddots & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{pmatrix}$
Vandermonde matrices $\left(t_i^j\right)_{i,j=0}^{n-1}$	**Cauchy matrices** $\left(\frac{1}{s_i-t_j}\right)_{i,j=0}^{n-1}$
$\begin{pmatrix} 1 & t_0 & \cdots & t_0^{n-1} \\ 1 & t_1 & \cdots & t_1^{n-1} \\ \vdots & & & \vdots \\ 1 & t_{n-1} & \cdots & t_{n-1}^{n-1} \end{pmatrix}$	$\begin{pmatrix} \frac{1}{s_0-t_0} & \cdots & \frac{1}{s_0-t_{n-1}} \\ \frac{1}{s_1-t_0} & \cdots & \frac{1}{s_1-t_{n-1}} \\ \vdots & & \vdots \\ \frac{1}{s_{n-1}-t_0} & \cdots & \frac{1}{s_{n-1}-t_{n-1}} \end{pmatrix}$

3 RC Processes for Structured Matrices

Extensions of unscaled RC processes (2.8), (2.3) to Toeplitz-like matrices can be found in [P92], [PBRZ99, Section 7.4]. Let us next follow [P01a] to outline these extensions in a unified way–simultaneously to various classes of structured matrices, including Toeplitz, Hankel, Vandermonde, and Cauchy matrices (see Table 3) and the matrices with the structures of these four types. This covers the most popular classes of structured matrices.

3.1 Structured Matrices and The Displacement Rank Approach

With a pair of $n \times n$ *operator matrices* A and B we associate linear *displacement operators* L, of Sylvester type $L = \nabla_{A,B}$,

$$\nabla_{A,B}(M) = AM - MB \tag{3.1}$$

and Stein type $L = \Delta_{A,B}$,

$$\Delta_{A,B}(M) = M - AMB \tag{3.2}$$

Table 3.2: Parameter and flop count for matrix representation and multiplication by a vector

Matrices M	Number of parameters per an $m \times n$ matrix M	Number of flops for computation of $M\mathbf{v}$
general	mn	$2mn - n$
Toeplitz	$m + n - 1$	$O((m+n)\log(m+n))$
Hankel	$m + n - 1$	$O((m+n)\log(m+n))$
Vandermonde	m	$O((m+n)\log^2(m+n))$
Cauchy	$m + n$	$O((m+n)\log^2(m+n))$

where M is an $n \times n$ matrix.

The operator matrix pairs $A, B \in \{D(\mathbf{s}), D(\mathbf{t}), Z_e, Z_f^T\}$, for appropriate vectors $\mathbf{s}$ and $\mathbf{t}$ and scalars e and f, cover the four cited most popular classes of structured matrices. The most used displacement operators satisfy the following properties:

a) the displacement $L(M)$ is a matrix having a small rank r for a structured matrix M and an associated displacement operator L (r is called the *displacement rank* of the matrix M),

b) the operator L^{-1} is linear, furthermore there are simple expressions for the matrix $M = L^{-1}(L(M))$ through its displacement $L(M)$, and

c) an $n \times n$ structured matrix can be multiplied by a vector fast: $O(nr\log^d n)$ flops for $d \leq 2$ (cf. Table 3.2).

In particular, for the operators $L_+ = \Delta_{Z,Z^T}$ and $L_- = \Delta_{Z^T,Z}$, it was proved in the seminal paper [KKM79] that the matrix equations

$$L(M) = GH^T,\ G = (\mathbf{g}_1, \ldots, \mathbf{g}_r),\ H = (\mathbf{h}_1, \ldots, \mathbf{h}_r) \tag{3.3}$$

imply that

$$M = \sum_{j=1}^{r} Z(\mathbf{g}_j) Z^T(\mathbf{h}_j) \tag{3.4}$$

for $L = L_+$ and

$$M = \sum_{j=1}^{r} Z^T(J\mathbf{g}_j) Z(J\mathbf{h}_j) \tag{3.5}$$

for $L = L_-$. It is easy to observe that

$$|\operatorname{rank}(L_+(M)) - \operatorname{rank}(L_-(M))| \leq 2,$$

for any matrix M, and that

$$\operatorname{rank}(L_+(M)) \leq 2, \ \operatorname{rank}(L_-(M)) \leq 2$$

where M is a Toeplitz matrix. This motivated the definition of *Toeplitz-like matrices* M as the ones with displacements $L_+(M)$ and $L_-(M)$ having small ranks. Expressions (3.4), (3.5) enable multiplications of a matrix M by a vector in $O(rn \log n)$ flops.

Similar simple expressions have been obtained in the case of displacement operators L associated with matrices of Hankel, Vandermonde, and Cauchy types [HR84], [BP94], [GO94], [PWa], [P01a], enabling *compressed representations* of an $n \times n$ structured matrix M via $2nr$ entries of the matrices G and H where $r = \operatorname{rank}(L(M))$. Orthogonal representations (3.3) for a given matrix $L(M)$ can be immediately obtained from its SVD [P92], [P93] (e.g., in the real case, $L(M) = U\Sigma^2 V^T$, $U^T U = V^T V = I_r$, $G = U\Sigma$, $H = V\Sigma$) and if $L(M)$ is a Hermitian matrix then from its eigendecomposition as well.

Compressed representations can be also derived based on some singular displacement operators. For instance, in [PBRZ99] the following known representation of an $n \times n$ Toeplitz-like matrix has been exploited,

$$M = Z_{f,lc}(M\mathbf{e}_{n-1}) + \frac{e}{e-f} \sum_{j=1}^{r} Z_f(Z_f \mathbf{g}_j) Z_{1/e}^T(\mathbf{h}_j) \tag{3.6}$$

Table 3.3: Some pairs of operators $\nabla_{A,B}$ and structured matrices

operator matrices		class of structured	rank of
A	B	matrices M	$\nabla_{A,B}(M)$
Z_1	Z_0	Toeplitz and its inverse	≤ 2
Z_1	Z_0^T	Hankel and its inverse	≤ 2
$Z_0 + Z_0^T$	$Z_0 + Z_0^T$	Toeplitz+Hankel	≤ 4
$D(\mathbf{t})$	Z_0	Vandermonde	≤ 1
Z_0	$D(\mathbf{t})$	inverse of Vandermonde	≤ 1
Z_0^T	$D(\mathbf{t})$	transposed Vandermonde	≤ 1
$D(\mathbf{s})$	$D(\mathbf{t})$	Cauchy	≤ 1
$D(\mathbf{t})$	$D(\mathbf{s})$	inverse of Cauchy	≤ 1

provided that (3.3) holds for $L = \nabla_{Z_f^{-1},Z_f^{-1}}$, where e and f are two scalars, $e \neq f$, $ef \neq 0$, and $Z_{f,lc}(\mathbf{v})$ denotes the f-circulant matrix of size $n \times n$ with the last column $\mathbf{v}$. (Note that $Z_f^{-1} = Z_{1/f}^T$.) Table 4.3 shows some displacement operators associated with structured matrices.

According to *the displacement rank approach,* one should operate with structured matrices M represented in a compressed form such as (3.3)–(3.6) and when required, recover the output (such as the solution of a linear system of equations) based on their linear expressions via the displacement $L(M)$. The entire approach can be represented by the following flowchart:

$$\text{COMPRESS} \longrightarrow \text{OPERATE} \longrightarrow \text{DECOMPRESS}.$$

At the OPERATE stage, the following simple results can be used [P01a].

Theorem 3.1. *For any linear operator L (in particular, for $L = \nabla_{A,B}$ and $L = \Delta_{A,B}$, for any pair of matrices A and B) and any pair of scalars a and b, we have $L(aM + bN) = aL(M) + bL(N)$.*

Theorem 3.2. *For any 5-tuple $\{A, B, C, M, N\}$ of $n \times n$ matrices, we have*

$$\nabla_{A,C}(MN) = \nabla_{A,B}(M)N + M\nabla_{B,C}(N),$$
$$\Delta_{A,C}(MN) = \Delta_{A,B}(M)N + AM\nabla_{B,C}(N).$$

Furthermore,

$$\Delta_{A,C}(MN) = \Delta_{A,B}(M)N + AMB\Delta_{B^{-1},C}(N),$$

if B is a non-singular matrix, whereas

$$\Delta_{A,C}(MN) = \Delta_{A,B}(M)N - AM\Delta_{B,C^{-1}}(N)C,$$

if C is a non-singular matrix.

Theorem 3.3. *Let M be a non-singular matrix. Then*

$$\nabla_{B,A}(M^{-1}) = -M^{-1}\nabla_{A,B}(M)M^{-1}.$$

Furthermore,

$$\Delta_{B,A}(M^{-1}) = BM^{-1}\Delta_{A,B}(M)B^{-1}M^{-1}$$

if B is a non-singular matrix, whereas

$$\Delta_{B,A}(M^{-1}) = M^{-1}A^{-1}\Delta_{A,B}(M)M^{-1}A$$

if A is a non-singular matrix.

3.2 Structured RC Processes

Based on the latter results and properties a)–c) of structured matrices listed in the previous subsection, one may perform structured matrix multiplication fast. $O(qnr^2 \log^d n)$ flops are sufficient per an RC step (2.1). This step outputs a short displacement generator of the matrix X_{i+1}, provided that the matrices M and X_i are given in compressed form (3.3) and q is the order of convergence of a process

(2.1). Special care is required, however, to contain the growth of rank$(L(X_{i+1}))$. With no care the rank rapidly increases; it may be tripled already in each Newton's step (2.8). Thus processes (2.1) should be modified as follows where the input matrix M is structured:

$$X_{i+1} = X(Y_{i+1}), \quad Y_{i+1} = c_{i+1} \sum_{k=0}^{p-1} R_i^k X_i \tag{3.7}$$

for R_i of (2.2). Here, the matrix $X_{i+1} = X(Y_{i+1})$ approximates the matrices Y_{i+1} and M^{-1}, and $r_{i+1} = \text{rank}(L(X_{i+1}))$ either equals or only slightly exceeds r. To complete the definition of the structured RC process (3.7) for fixed parameters p, c_{i+1}, let us specify the transition from the matrix Y_{i+1} to the matrix X_{i+1}, where both structured matrices Y_{i+1} and X_{i+1} are represented by their displacements [P92], [P92a], [BP93], [PZHD97], [PBRZ99], [PR01], [PRW01].

Approach I. Truncation of the smallest singular values of the displacement. Compute the SVD of $L(Y_{i+1}) = G_{i+1}H_{i+1}^T$ (cf. [HLPW86]) and truncate the smallest singular values to obtain a displacement matrix $L(X_{i+1})$ having r_{i+1} (non-zero) singular values, where r_{i+1} is fixed according to a selected policy, say $r_{i+1} \leq r$ or $r_{i+1} \leq cr$ for a fixed constant c. (In the case where $L(X_i)$ is a Hermitian matrix, one may rely on its eigendecomposition instead of its SVD.)

Approach II. Substitution of a computed approximation for the inverse in the inversion formulae. Compute the displacement $L(X_{i+1})$ based on Theorem 3.3, where M^{-1} is replaced by X_i. That is, for $S_{p,i} = S_i$ of (3.7),

$\tilde{S}_{p,i} = c_{i+1}\sum_{k=o}^{p-1}\tilde{R}_i^k$, $\tilde{R}_i = I - MX_i$, Y_{i+1} of (4.7), $i = 0, 1, \ldots$, write $L(M) = GH^T$,

$$\begin{aligned} L(X_{i+1}) &= \nabla_{B,A}(X_{i+1}) \\ &= G_{i+1}H_{i+1}^T \\ &= -Y_{i+1}\nabla_{A,B}(M)Y_{i+1} \\ &= (-Y_{i+1}G)(H^TY_{i+1}) \\ &= -S_{p,i}(X_iG)(H^TX_i)\tilde{S}_{p,i}; \end{aligned} \tag{3.8}$$

also write either

$$\begin{aligned} A^{-1}L(X_{i+1}) &= A^{-1}\Delta_{B,A}(X_{i+1}) \\ &= A^{-1}G_{i+1}H_{i+1}^TA \\ &= A^{-1}Y_{i+1}A^{-1}\Delta_{A,B}(M)Y_{i+1}A \\ &= A^{-1}(Y_{i+1}A^{-1}G)(H^TY_{i+1}A) \\ &= A^{-1}S_{p,i}(X_iA^{-1}G)(H^TX_i)\tilde{S}_{p,i}A \end{aligned} \tag{3.9}$$

where the operator matrix A is non-singular or

$$\begin{aligned} L(X_{i+1})B^{-1} &= \Delta_{B,A}(X_{i+1})B^{-1} \\ &= BG_{i+1}H_{i+1}^TB^{-1} \\ &= BY_{i+1}\Delta_{A,B}(M)B^{-1}Y_{i+1}B^{-1} \\ &= (BY_{i+1}G)(H^TB^{-1}Y_{i+1})B^{-1} \\ &= BS_{p,i}(X_iG)(H^TB^{-1}X_i)\tilde{S}_{p,i}B^{-1} \end{aligned} \tag{3.10}$$

where the operator matrix B is non-singular. The computation of the displacement of X_{i+1} in (3.8)–(3.10) essentially amounts to post-multiplying $S_{p,i}$ by X_iG or $X_iA^{-1}G$ and either post-multiplying H^TX_i by $\tilde{S}_{p,i}$ or $\tilde{S}_{p,i}A$ or pre-multiplying $\tilde{S}_{p,i}B^{-1}$ by X_i and the product by H^TB^{-1}. In each case, we multiply each of the matrices R_i and R_i^T by $O(pr)$ vectors. RC process (3.8)–(3.10) can be applied to a

Toeplitz matrix M. Then we would multiply each of the matrices X_i, M, M^T, and X_i^T by two vectors for every i, whereas the RC process (3.7), (3.8) only requires multiplication of each of M and X_i by a pair of vectors for every i. Furthermore, since we assume that $X_i \approx Y_i$, we may replace $-X_iG$ by $G_i = -Y_iG$ and H^TX_i by $H_i^T = H^TY_i$ in (3.8) and thus to replace (3.8) by a simpler expression

$$L(X_{i+1}) = G_{i+1}H_{i+1}^T = S_{p,i}G_iH_i^T\tilde{S}_{p,i} \tag{3.11}$$

Similarly, we may simplify (3.9) and (3.10) to write

$$A^{-1}L(X_{i+1}) = A^{-1}G_{i+1}H_{i+1}^T,$$

$$A^{-1}G_{i+1} = A^{-1}Y_{i+1}A^{-1}G = A^{-1}S_{p,i}G_i,$$

$$H_{i+1}^T = H^TY_{i+1} = H_i^T\tilde{S}_{p,i}$$

where A is nonsingular and to write

$$L(X_{i+1})B^{-1} = G_{i+1}H_{i+1}^TB^{-1},$$

$$G_{i+1} = Y_{i+1}G = S_{p,i}G_i,$$

$$H_{i+1}^TB^{-1} = H^TB^{-1}Y_{i+1}B^{-1} = H_i^T\tilde{S}_{p,i}B^{-1}$$

where B is nonsingular.

Approach I relies on the observation that

$$||L(X_{i+1}) - L(Y_{i+1})|| \le ||L(X_{i+1}) - L(M^{-1})||$$

under the 2-norm and the Frobenius norm. This observation is due to Theorem 3.3 and to the well-known results on the lower rank matrix approximation based on the truncation of the singular values [GL96]. Thus we bound the norms $||L(X_{i+1}) - L(M^{-1})||$ and $||X_{i+1} - M^{-1}|| \le ||L^{-1}||\,||L(X_{i+1}) - L(M^{-1})||$ in terms of the norm $||L(Y_{i+1}) - L(M^{-1})||$.

In Approach II, we bound the same norms by combining (3.8)–(3.10) with Theorem 3.3.

Specific estimates for the approximation errors, the convergence rate, and the initial residual or error norms which ensure rapid convergence for both approaches can be found in [P92], [PZHD97], [PBRZ99], [PRW01], and [P01a].

Algorithm 3, [PBRZ99, Algorithm 7.4.1], applies Approach I to Toeplitz-like matrices M and uses the displacements $L_+(M)$ and $L_-(X_i)$ and expressions (3.4), (3.5) to compute the displacements $L_-(X_{i+1}) = L_-(X(Y_{i+1}))$. It is proved in [PBRZ99] that in this case

$$||X_{i+1} - M^{-1}||_2 \le (1 + 2(r_i - r)n)||X_i - M^{-1}||_2 \tag{3.12}$$

where $r_i = \text{rank}(L_-(Y_i))$.

Algorithm 4, [PBRZ99, Algorithm 7.4.2], implements Approach II and relies on (2.3), (2.8), and (3.6). In this case the matrix X_{i+1} is defined by its displacement

$$\nabla_{Z_f^{-1},Z_f^{-1}}(X_{i+1}) = G_{i+1}H_{i+1}^T,$$

$$G_{i+1} = X_i(2I - MX_i)G, \quad H_{i+1}^T = H^T X_i(2I - MX_i)$$

and by its last column

$$X_{i+1}\mathbf{e}_{n-1} = (2I - MX_i)X_i\mathbf{e}_{n-1}.$$

In [PRW01] both Approaches I and II have been elaborated upon and analyzed in a unified way for various classes of structured matrices (based on the displacement rank approach). The results of [SS74] and [PS91] on the convergence of Newton's and other RC processes in Section 2 do not apply to processes (3.7) because of the compression of the displacements $L(Y_i)$, but Remark 3.2 outlines some methods of extension to both Approaches I and II.

The following theorems from [PRW01] (extending their preliminary versions of [P92], [PZHD97], [PBRZ99], and [PR01]) state the estimates for the error norms of the computed approximations under assumption (2.3). The error norms grow proportionally to the norm $||L^{-1}||_l$ of the inverse of the displacement operator L,

$$||L^{-1}||_l = \sup_M(||M||_l/||L(M)||_l), \quad l = 1, 2, \infty.$$

Upper estimates for this norm, $||L^{-1}||_l$ for various customary operators L associated with the most popular classes of structured matrices have been deduced in [PRW01] and [PWb] (see also [P01]).

Theorem 3.4. *[PRW01]. Let the unscaled Newton-Structured process (2.8), (2.3) be applied to a non-singular matrix M. Let all its steps be performed with compression according to (3.7) and Approach I such that all singular values of the displacements $L(Y_i)$, except for the r largest ones were truncated where $r = \mathrm{rank}(L(M^{-1}))$. Then we have*

$$||X_i - M^{-1}||_2 \le ||I - X_iM||_2 \, ||M^{-1}||_2 \le \theta^{2^i}||M^{-1}||_2/\eta,$$

$i = 1, 2, \ldots,$ provided that

$$\theta = ||I - X_0M||_2\eta,$$

$$\eta = (1 + (||A||_2 + ||B||_2)||L^{-1}||_2)\sigma_1(M)/\sigma_n(M) \quad \text{for } L = \nabla_{A,B},$$

$$\eta = (1 + (1 + ||A||_2||B||_2)||L^{-1}||_2)\sigma_1(M)/\sigma_n(M) \quad \text{for } L = \Delta_{A,B}.$$

Theorem 3.5. *[PRW01]. Let the unscaled Newton-Structured process (2.3), (2.8) be applied to invert a non-singular matrix M. Let (3.7) and Approach II be used for the compression of the displacements $L(Y_i)$, $i = 1, 2, \ldots$. Write*

$$r_{i,l} = ||I - X_iM||_l,$$

$$e_{i,l} = ||Y_i - M^{-1}||_l,$$

$$\hat{e}_{i,l} = ||X_i - M^{-1}||_l,$$

$$l = 1, 2, \infty;\ i = 0, 1, 2, \ldots$$

Let $r_0 \le 1$, $e_{i,l} \le ||M^{-1}||_l$, $l = 1, 2, \infty$; $i = 0, 1, 2, \ldots$,

$$C_l = 3||L^{-1}||_l\ ||L(M)||_l\ ||X_0||_l/(1 - r_{0,l}) \text{ for } L = \nabla_{A,B},$$

$$C_l = 3||L^{-1}||_l\ ||L(M)||_l\ ||M||_l\ ||M^{-1}||_l\ ||X_0||_l/(1 - r_{0,l}) \text{ for } L = \Delta_{A,B}.$$

Then

$$\hat{e}_{i,l} \le C_l e_{i,l}, \quad e_{i+1,l} \le (C_l e_{i,l})^2 ||M||_l,$$

and therefore,

$$\gamma_l e_{i+1,l} \le (\gamma_l e_{1,l})^{2^i}, \quad i = 1, 2, \ldots;\ l = 1, 2, \infty,$$

where $\gamma_l = C_l^2 ||M||_l$.

Algorithm 4, [PBRZ99, Algorithm 7.4.2] can be modified according to (3.11) as follows:

$$\mathbf{x}_j = X_j \mathbf{e}_j,$$

$$\nabla_{Z_f^{-1}, Z_f^{-1}}(X_j) = G_j H_j^T, \quad j = 0, 1, \ldots;$$

$$G_{i+1} = (2I - MX_i)G_i,$$

$$H_{i+1}^T = H_i^T(2I - MX_i),$$

$$\mathbf{x}_{i+1} = (2I - MX_i)\mathbf{x}_i, \quad i = 0, 1, \ldots,$$

saving multiplication of r vectors by X_i and X_i by r vectors in each iteration step i.

Remark 3.1. *Newton-Structured Iteration with compression was first studied for Toeplitz-like matrices in [P92]. In [PR01], [PRW01] the algorithms were extended to various other classes of structured matrices in a unified way, adopted in this section. In an alternative* displacement transformation approach *due to [P90], it was*

proposed to extend successful algorithms available for one class of structured matrices to various other classes by means of the transformation of the associated displacement operators; furthermore, sample displacement transformation techniques were shown for the transformation in all directions among the operators associated with the matrices having structures of Toeplitz, Hankel, Vandermonde, and Cauchy types. In particular, these techniques apply to matrix inversion and thus enable immediate extension of our RC and HRC processes. So far, the most acclaimed application of the displacement transformation approach has been the reduction of the practical solution of Toeplitz and Toeplitz-like linear systems of equations to the Cauchy-like case via the transformation of the associated displacement operators [H95], [GKO95].

Remark 3.2. *Applying both Approaches I and II we may try to improve convergence by allowing more work per iteration step and using processes with the matrices $S_{p,i}$ and $\tilde{S}_{p,i}$ of (3.8)–(3.12) for larger p_i, replacing matrices $S_{p,i}$ and $\tilde{S}_{p,i}$ by $p_i(MX_i)$ for selected polynomials $p_i(y)$ (compare(2.14)) or generalizing the approaches of (3.9) and (3.10). Approach I also allows us to vary the level of compression by truncating more or fewer singular values of $L(X_i)$. We call the respective modifications* Structured RC processes with delayed compression.

Remark 3.3. *For a more general compression policy extending both Approaches I and II, see [CPVBWa].*

4 Toeplitz-Like Matrices

4.1 Some Definitions and Basic Results

When A and X_0 are structured matrices, say, Toeplitz or Toeplitz-like matrices, the computations required for Newton's iteration can be carried out more efficiently. Following the discussion in [KKM79], we shall say that a matrix $R \in \mathbf{C}^{n\times n}$ is *Toeplitz-like* if its displacements

$$\nabla_-(R) = R - Z^T RZ \quad \text{and/or} \quad \nabla_+(R) = R - ZRZ^T$$

have a small rank, say, r, where $r << n$, and

$$Z = \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}, \quad Z = (z_{i,j}),\ z_{i,j} = 1 \text{ if } i+1 = j,\ z_{i,j} = 0 \text{ otherwise.}$$

Here and hereafter, W^T denotes the transpose of a matrix or a vector W.

We have

$$\nabla_-(R) = R - Z^T RZ = G_-H_-^T \quad \text{and/or} \quad \nabla_+(R) = R - ZRZ^T = G_+H_+^T$$

for some matrices $G_-, H_-, G_+, H_+ \in \mathbf{R}^{n\times r}$. The pairs of matrices G_-, H_- and G_+, H_+ will be called *generators of length* r for $\nabla_-(R)$ and $\nabla_+(R)$, respectively, as well as *displacement generators of length* r for R.

We shall use the following basic facts (established in [KKM79]).

Lemma 4.1. *[Basic Properties of Structured Matrices] Given G, H in $\mathbf{R}^{n\times r}$, there exists a unique matrix T_+ and a unique matrix T_- satisfying*

$$\begin{aligned}
\nabla_+(T_+) &= T_+ - ZT_+Z^T &= GH^T\ ,\\
\nabla_-(T_-) &= T_- - Z^TT_-Z &= GH^T\ ,\\
T_+ &= \sum_{i=1}^{r} L(\mathbf{g}_i)L^T(\mathbf{h}_i)\ ,\\
T_- &= \sum_{i=1}^{r} L^T(J\mathbf{g}_i)L(J\mathbf{h}_i)\ ,
\end{aligned}$$

where $\mathbf{g}_i$ and $\mathbf{h}_i$ denote the i-th columns of G and H, respectively, and $L(\mathbf{v})$ denotes a lower triangular Toeplitz matrix with the first column given by vector $\mathbf{v}$.

Lemma 4.2. *The operators ∇_- and ∇_+ satisfy the following properties for any non-singular matrix R and for any pair of $n\times n$ matrices X and Y:*

(a) $\operatorname{rank}(\nabla_+(R)) = \operatorname{rank}(\nabla_-(R^{-1}))$.

(b) $\operatorname{rank}(\nabla_-(XY)) \le \operatorname{rank}(\nabla_-(X)) + \operatorname{rank}(\nabla_-(Y)) + 1$.

(c) Given two pairs of generator matrices of lengths a and b for $\nabla_-(X)$ and $\nabla_-(Y)$, respectively, a pair of generator matrices of length at most $ab+1$ for $\nabla_-(XY)$ can be computed by using $O(nab\log n)$ flops.

(d) Given a pair of generator matrices of length r for $\nabla_+(X)$, a pair of generator matrices of length $\bar{r} \le r+2$ for $\nabla_-(X)$ can be computed by using $O(nr\log n)$ flops.

Proof.

(a), (d). See [KKM79], [BP93, pp. 176, 178].

(b), (c). We have

$$\begin{aligned}
\nabla_-(X) &= X - Z^T X Z = G_X H_X^T , \\
\nabla_-(Y) &= Y - Z^T Y Z = G_Y H_Y^T , \\
\nabla_-(XY) &= XY - Z^T XYZ \\
&= XY - Z^T XIYZ \\
&= XY - Z^T XZY + Z^T XZY \\
&\qquad -Z^T XZZ^T YZ + Z^T XZZ^T YZ - Z^T XIYZ \\
&= \nabla_-(X)Y + Z^T XZ\nabla_-(Y) + Z^T X(ZZ^T - I)YZ .
\end{aligned}$$

We also observe that $ZZ^T - I = -\mathbf{e}^{(0)}\mathbf{e}^{(0)T}$, where $\mathbf{e}^{(0)T} = [1, 0, \ldots, 0]^T$ is a unit coordinate vector. Therefore, $\nabla_-(XY) = G_{XY} H_{XY}^T$, where the pair of 1×3 block matrices

$$G_{XY} = [\, G_X \,,\, Z^T XZG_Y \,,\, -Z^T X\mathbf{e}^{(0)} \,] , \quad H_{XY} = [\, Y^T H_X \,,\, H_Y \,,\, Z^T Y^T \mathbf{e}^{(0)} \,]$$

is an explicit expression of the generator (G_{XY}, H_{XY}) of the displacement $\nabla_-(XY)$ via the three generators:

- (G_Y, H_Y) of the displacement $\nabla_-(X)$,
- (G_X, H_X) of the displacement $\nabla_-(Y)$, and
- $(-Z^T X\mathbf{e}^{(0)}, Z^T Y^T \mathbf{e}^{(0)})$ of the rank-1-matrix $Z^T X(ZZ^T - I)YZ$.

Such expressions imply part (b) of Lemma 4.2. To complete the proof of part (c), observe that the computation of G_{XY} and H_{XY} essentially amounts to the multiplications of X by ZG_Y and of Y^T by H_X and then estimate the computational cost of these multiplications by applying Lemma 4.1 and recalling that an $n \times n$ Toeplitz matrix can be multiplied by a vector in $O(n \log n)$ flops. □

5 Newton-Toeplitz Iteration

5.1 The First Version (Outline)

Let us describe Newton-Toeplitz iteration, that is, Newton's iteration (1.1) in the Toeplitz-like case. Until section 5.4, we shall assume that A and X_0 are structured matrices with $\mathrm{rank}(\nabla_+(A)) = \mathrm{rank}(\nabla_-(A^{-1})) = r$, $\mathrm{rank}(\nabla_-(X_0) = r_0$ and that the matrices A and X_0 are given with their ∇_--generators of the minimum length, $\bar{r}$ and r_0, respectively, where $\bar{r} \leq r + 2$ by Lemma 4.2(d), r and r_0 are small relatively to n. Then, in view of the above results, a displacement generator for X_k having length at most $\bar{r}_k = 2^k r_0 + (2^k - 1)(\bar{r} + 3)$ can be computed at the overall cost of $O(n(\bar{r}_{k-1} + \bar{r})\bar{r}_{k-1} \log\ n)$ flops, $k = 1, 2, \ldots$. The values $\bar{r}_k$ and $\sum_{i=0}^{k-1}(\bar{r}_i + \bar{r})\bar{r}_i$ have orders of 2^k and 4^k, respectively, so that the iteration (1.1) soon becomes quite costly.

In some cases, a few steps (1.1) may already suffice for the refinement of a given approximation X_0 to A^{-1}, and then the overall computational cost is small enough. If one needs many steps (1.1), then the techniques of the next two sections will help us to control the growth of the generator length and the computational cost.

Let us outline these techniques. By assumption, $\mathrm{rank}(\nabla_-(A^{-1})) = r$. Hence, the matrices $-X_i$, which approximate A^{-1} closely for larger i, have a nearby matrix of ∇_--displacement rank r for larger i.

This fact suggests the following approach to decreasing the computational cost of the iterative scheme (1.1): shift from X_i to a nearby matrix Y_i having displacement rank at most r and then restart the iteration with Y_i instead of X_i.

The advantage of this modification is the decrease of the computational cost at the i-th Newton's step and at all the subsequent steps. The disadvantage is a possible deterioration of the approximation, since we do not know A^{-1}, and

the transition from X_i to Y_i may occur in a wrong direction. Both X_i and Y_i, however, lie close to A^{-1} and, therefore, to each other. The latter observation enables us to bound the deterioration of the approximation relatively to the current approximation error, so that the quadratic improvement in the error bound at the next step of Newton's iteration will immediately compensate us for such a deterioration. Moreover, the transition from X_i to a nearby matrix Y_i of a small displacement rank can be done at a low computational cost. We will supply more details in the next two sections.

5.2 Bounding Displacement Rank

To estimate the errors of the approximations of A^{-1} by X_i and Y_i, we first recall the following basic result (cf. [GL96, pp.72, 230]).

Lemma 5.1. *Given a matrix W of rank r, it holds that*

$$\sigma_r = min_{rank\ (B)\le r}||W - B||_2,$$

that is, the error in the optimal choice of an approximant of W whose rank does not exceed r is equal to the r-th singular value of W. Moreover, the condition number of W is equal to the ratio σ_1/σ_n.

The reader may assume below that the numbers r and r_0 are small ($\overline{r} = r = 2$ for a Toeplitz matrix A) and that r_i is not much larger than r, say, $r_i \le 3r + 3$. Hereafter, until section 5.4, we will use the displacement $\nabla_-(R) = R - Z^T RZ$.

Algorithm 1, *Bounding the Displacement Rank of Newton's Iterates.*
Input: A positive integer $r = \text{rank}(\nabla_+(A)) = \text{rank}(\nabla_-(A^{-1}))$ and a displacement generator G_i, H_i, of length at most $r_i \ge r$, for a matrix X_i, such that $\nabla_-(X_i) = G_i H_i^T$.

Output: A displacement generator of length at most r for a matrix Y_i such that

$$||Y_i + A^{-1}||_2 \leq q(1 + 2n(r_i - r))||X_i + A^{-1}||_2.$$

Computations:

1. Compute the SVD of the displacement $\nabla_-(X_i) = U_i\Sigma_i V_i^T$. (The computation is not costly (cf. [P93]) since it is performed for $G_iH_i^T$, where $G_i, H_i \in C^{n\times r_i}$, and since r_i is small.)

2. Set to zero the $r_i - r$ smallest singular values of $\nabla_-(X_i)$ in the matrix in Σ_i, thus turning Σ_i into a diagonal matrix of rank r.

3. Compute and output the matrices $\bar{G}_i$ and $\bar{H}_i$ obtained from the matrices $U_i\Sigma_i$ and V_i, respectively, by deleting their last $r_i - r$ columns.

The overall computational cost of this algorithm is $O(n \log\ n)$ flops. Of course, this covers only the computation by Algorithm 1, which is just part of a single step of Newton's iteration. (cf. (5.1) below).

Now let $Y_i \in \mathbf{C}^{n\times n}$ denote the unique matrix defined by its ∇_--generator $\bar{G}_i, \bar{H}_i$. Correctness of the algorithm is implied by the following result of [P92], [P93], and [P93a], which shows that the matrix Y_i approximates A^{-1} almost as well as X_i does.

Theorem 5.1. *It holds that* $||Y_i + A^{-1}||_2 \leq (1 + 2(r_i - r)n)||X_i + A^{-1}||_2$.

Outline of the proof: The proof of this basic result was given in [P92], [P93], and [P93a], and relies on the following observations. The matrices A^{-1} and $-X_i$ closely approximate each other; therefore, so also do their ∇_--displacements, $\nabla_-(A^{-1})$ and $\nabla_-(X_i)$. Since the approximation of $\nabla_-(X_i)$ by $\nabla_-(Y_i)$ is optimal by Lemma 5.1, we have $||\nabla_-(Y_i) - \nabla_-(X_i)||_2 \leq ||\nabla_-(A^{-1}) + \nabla_-(X_i)||_2$. The map from the

∇_--displacements, $\nabla_-(Y_i), \nabla_-(X_i), \nabla_-(A^{-1})$, back to the matrices Y_i, X_i, A^{-1} may change this bound by at most factor of $1 + 2(r_i - r)n$. □

We may now modify Newton's iteration (1.1) by incorporating the construction of Algorithm 1.

Let a matrix A and an initial approximation X_0 of A^{-1} be given with their ∇_--generators of length $\bar{r} \leq r + 2$ and r_0, respectively. We recursively compute the matrices $Y_0, X_1, Y_1, X_2, \ldots$, satisfying

$$X_{i+1} = Y_i(2I + AY_i) = (2I + Y_iA)Y_i, \ i = 0, 1, \ldots, \tag{5.1}$$

where Y_i denotes the output matrix of Algorithm 1 applied to X_i. Thus, for each i, we first apply Algorithm 1 to the matrix X_i followed by the computation of the matrix X_{i+1} based on (5.1). Since Y_i is a Toeplitz-like matrix represented with its displacement generator of length at most r, the overall computational cost is $O(nr^2 \log\ n)$ for each step of (5.3).

5.3 Convergence Rate and Complexity Estimates

Define

$$p(i) = ||I + Y_iA||_2, \ i = 0, 1, \ldots, \tag{5.2}$$

and deduce from (5.1) that $I + X_{i+1}A = (I + Y_iA)^2$, so that

$$||I + X_{i+1}A||_2 \leq p^2(i), \ ||A^{-1} + X_{i+1}||_2 \leq p^2(i)||A^{-1}||_2 \ .$$

Here and hereafter, we write $p^h(i)$ to denote $(p(i))^h$. By Thm. 5.1, we obtain

$$||A^{-1} + Y_{i+1}||_2 \ \leq \ (1 + 2(r_i - r)n)||A^{-1} + X_{i+1}||_2 \ \leq \ (1 + 2(r_i - r)n)p^2(i)||A^{-1}||_2,$$

and hence

$$p(i+1) \ = \ ||I + Y_{i+1}A||_2 \ \leq \ ||A^{-1} + Y_{i+1}||_2||A||_2$$

$$\leq (1+2(r_i-r)n)p^2(i)||A^{-1}||_2||A||_2 \;=\; (1+2(r_i-r)n)\mathrm{cond}_2(A)p^2(i)\;.$$

Therefore, for a positive b, the inequality

$$(1+2(r_i-r)n)\mathrm{cond}_2(A)p^{1-b}(i) \;\leq\; 1\;,\ \text{for all } i, \tag{5.3}$$

implies a convergence rate of $1+b$, that is, it implies the bound

$$p(i+1) \;\leq\; p^{1+b}(i)\;,\ \text{for all } i. \tag{5.4}$$

By observing that rank $\nabla_-(R+I) \leq 1 +$ rank $\nabla_-(R)$ for any matrix R and by applying Lemma 4.2(c) to the matrices of equation (5.1), we obtain that

$$r_i - r \;\leq\; r+\bar{r}+3\;,\ \bar{r} \;\leq\; r+2\;,$$

where the length of the ∇_--generators of Y_i and A is at most r and $\bar{r}$, respectively. If $p(i) \leq 1$ and $p(i)$ satisfies (5.3), then (5.4) implies that $p(i+1) \leq p(i)$. Thus it suffices to assume (5.3) with $p(i)$ replaced by $p(0)$ and $r_i - r$ replaced by $r+\bar{r}+3$. The results are summarized below (cf. [P92], [P93], and [P93a]).

Theorem 5.2. *Let rank* $(\nabla_-(A^{-1})) \leq r$ *and let the matrices* X_0 *and* A *be given with their* ∇_-*-displacement generators of length* $r_0 \leq r$ *and* $\bar{r} \leq r+2$*, respectively. Let*

$$(1+2(r+\bar{r}+3)n)\mathrm{cond}_2(A)p^{1-b}(0) \;\leq\; 1\;, \tag{5.5}$$

for p(0) of (5.2) (for i=0) and for some fixed positive $b<1$*. Then, for* $i = 0, 1, \ldots$*, we have*

$$\begin{aligned} \mathit{rank}\,(\nabla_-(Y_i)) \;&\leq\; r\;,\\ \mathit{rank}\,(\nabla_-(X_i)) \;&\leq\; 2r+\bar{r}+3\;,\\ ||X_i + A^{-1}||_2 \;=\; ||(-X_i)-A^{-1}||_2 \;&\leq\; (p(0))^{(1+b)^i}||A^{-1}||_2\;, \end{aligned}$$

and the matrices $Y_1,\ X_1,\ Y_2,\ X_2,\ \ldots\ ,\ Y_i,\ X_i$ *can be computed at the overall cost of performing* $O(inr^2 \log\, n)$ *flops.*

We can see that unless the matrix A has a very large condition number (that is, unless A is very ill-conditioned), (5.5) is a mild assumption on the residual norm $p(0)$. This assumption is sufficient to ensure a rapid improvement of the initial approximation of A^{-1} by $-X_0$ at a low computational cost of $O(inr^2 \log\ n)$ flops. Therefore, the bound $p_i = ||I + X_i A||_2 \leq \epsilon$ is ensured already in

$$i = \left\lceil \log_2 \left(\frac{\log p(0)}{\log \epsilon} \right) \right\rceil \tag{5.6}$$

Newton-Toeplitz steps (5.1).

Remark 5.1. *If the initial approximation X_0 does not satisfy the bound (7), various homotopy techniques can be a remedy. The most straightforward way is to apply our algorithm recursively to compute the matrices*

$$A_i^{-1} = (A_0 + t_i(A - A_0))^{-1}, \text{ for } i = 0, 1, \ldots, K,$$

where A_0 is a readily invertible matrix, $t_0 = 0$, $t_K = 1$, and a sufficiently large natural K and an increasing sequence $t_0, t_1, \ldots, t_K$ are chosen to ensure rapid convergence of our algorithm in each recursive step. Similar treatment can be applied to the other algorithms, and we refer the reader to [P92] on a particular application of homotopy techniques to the design of fast Toeplitz-like solvers.

5.4 f-Circulant Matrices and A Variant of Newton-Toeplitz Iteration

The displacement structure of Toeplitz-like matrices can be represented in terms of other operators, in particular, of the operator

$$\nabla_f(A) = A - Z_f A Z_{1/f}^T ,$$

where $f \neq 0$ is a scalar and Z_f is a unit f-circulant matrix,

$$Z_f = \begin{bmatrix} 0 & \cdots & 0 & f \\ 1 & 0 & & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & & 1 & 0 \end{bmatrix} = Z + f\mathbf{e}^{(0)}\mathbf{e}^{(n-1)T}.$$

Here and hereafter, $\mathbf{e}^{(0)} = [1, 0, \ldots, 0]^T$ and $\mathbf{e}^{(n-1)} = [0, \ldots, 0, 1]^T$ denote two unit coordinate vectors. (Circulant matrices are 1-circulant.)

Since $Z_f = Z + f\mathbf{e}^{(0)}\mathbf{e}^{(n-1)T}$, we have rank $(\nabla_f(A) - \nabla_-(A)) \leq 2$, rank $(\nabla_f(A) - \nabla_g(A)) \leq 2$ for any triple (A, f, g). Lemmas 4.1 and 4.2 are easily extended (cf. [GO92], [BP93], pp.182, 195), and so is Algorithm 1. In particular, we will use the next result.

Lemma 5.2. *If* $e \neq f$, $e \neq 0$ *(say,* $f = 1$, $e = -1$*) and if* $\nabla_f(A) = GH^T = \sum_{i=1}^r \mathbf{g}^{(i)}\mathbf{h}^{(i)T}$, *then we have*

$$A = Z_{f,lc}(A\mathbf{e}^{(n-1)}) + \frac{e}{e-f}\sum_{i=1}^{r} Z_f(\mathbf{g}^{(i)})Z_{1/e}^T(\mathbf{h}^{(i)}),$$

where $Z_f(\mathbf{v}) = \sum_{i=0}^{n-1} v_i Z_f^i$ *denotes the f-circulant matrix with the first column* $\mathbf{v} = [v_0, \ldots, v_{n-1}]^T$, *and* $Z_{f,lc}(\mathbf{v})$ *denotes the f-circulant matrix with the last column* $\mathbf{v}$.

The complication due to the extra term $Z_f(A\mathbf{e}^{(n-1)})$ in the representation of Lemma 5.2 (versus the one of Lemma 4.1) is compensated by the simplification of the multiplication of the matrices $Z_f(\mathbf{v})$ by a vector versus the same operation with the matrices $L(\mathbf{v})$ or $L^T(\mathbf{v})$. (The former multiplication requires roughly twice fewer flops than the latter.)

The matrices Z_f are nonsingular, and we may use the operators $\nabla^f(A) = Z_f^{-1}A - AZ_{1/f}^T$ instead of $\nabla_f(A)$. This enables a distinct version of Newton's iteration, which we will demonstrate next. (Note that a generator G, H for $\nabla_f(A)$) immediately turns into the generator $Z_f^{-1}G, H$ for $\nabla^f(A)$ and vice versa.) In particular, Lemma 5.2 turns into the following result.

Lemma 5.3. *If* $e \neq f$, $e \neq 0$ *(say,* $f = 1$, $e = -1$*) and if* $\nabla^f(A) = GH^T = \sum_{i=1}^r \mathbf{g}^{(i)}\mathbf{h}^{(i)T}$, *then we have*

$$A = Z_{f,lc}(A\mathbf{e}^{(n-1)})\frac{e}{e-f}\sum_{i=1}^{r} Z_f(Z_f(\mathbf{g}^{(i)}))Z_{1/e}^T(\mathbf{h}^{(i)}),$$

where $Z_f(\mathbf{v})$ and $Z_{f,lc}(\mathbf{v})$ are defined as in Lemma 5.2.

For our numerical tests, we chose $f = 1$ and $e = -1$, though other choices of f and e, with $f \neq e$, $e \neq 0$, are also possible. (Moreover, the operators ∇^f can be replaced by $\nabla^+(A) = ZA - AZ$ or $\nabla^-(A) = Z^TA - AZ^T$, to which Lemma 5.2 is easily extended, see [BP93], p. 184.)

Now suppose that $\nabla^f(A) = GH^T$, write $X = -A^{-1}$, and deduce that $\nabla^f(X) = G_X H_X^T = X\nabla^f(A)X = XGH^TX$, so that

$$G_X = XG \ , \ H_X^T = H^TX \ . \tag{5.7}$$

Indeed,

$$\begin{aligned} X\nabla^f(A)X &= XZ_f^{-1}AX - XAZ_{1/f}^TX \\ &= -XZ_f^{-1} + Z_{1/f}^TX \\ &= Z_f^{-1}X - XZ_{1/f}^T \\ &= \nabla^f(X) \end{aligned}$$

where the second last equation follows because $Z_{1/f}^T = Z_f^{-1}$.

Next, based on (5.7) and Lemmas 5.2 and 5.3, we will modify Algorithm 1. (To distinguish from Algorithm 1, we use the notation $\tilde{X}_i$ rather than X_i for the computed approximations to $X = -A^{-1}, i = 1, 2, \ldots$, and, as in the case of Algorithm 1, we assume that an initial approximation $\tilde{X}_0$ is given from outside, in this case in the form of the ∇^f-displacement generator matrices G_0, H_0 and the last column vector $\tilde{X}_0\mathbf{e}^{(n-1)}$.)

Algorithm 2, *an Alternative Method of Bounding the Displacement Rank of Newton's Iterates.*

Input: A natural $\tilde{r}$ and two matrices A and $\tilde{X}_i$ given with their ∇^f-displacement generators, $G = G_A$, $H = H_A$ and $G_i = G_{\tilde{X}_i}$, $H_i = H_{\tilde{X}_i}$, respectively, both of length at most $\tilde{r}$, and with their last columns, $A\mathbf{e}^{(n-1)}$ and $\tilde{X}_i\mathbf{e}^{(n-1)}$, respectively.

Output: A ∇^f-displacement generator G_{i+1}, H_{i+1} of length at most $\tilde{r}$ for a matrix $\tilde{X}_{i+1}$ and its last column $\tilde{X}_{i+1}\mathbf{e}^{(n-1)}$ satisfying

$$\tilde{p}_{i+1} = ||I + \tilde{X}_{i+1}A||_1 \leq \tilde{p}_i^2 \mathrm{cond}_1(A)(1 + 0.5r||G||_1||H^T||_1||X||_1(2 + \tilde{p}_i^2)),$$

$$\tilde{p}_i = ||I + \tilde{X}_i A||_1 \tag{5.8}$$

Computations: Apply Lemma 5.3 to express A and $\tilde{X}_i$ through the input parameters. Then compute and output the vector

$$\tilde{X}_{i+1}\mathbf{e}^{(n-1)} = \tilde{X}_i(2I + A\tilde{X}_i)\mathbf{e}^{(n-1)}$$

and the matrices

$$G_{i+1} = \tilde{X}_i(2I + A\tilde{X}_i)G,$$

$$H_{i+1}^T = H^T\tilde{X}_i(2I + A\tilde{X}_i).$$

Lemma 5.3 reduces the computations by Algorithm 2 to a sequence of $6\tilde{r}(2\tilde{r}+1)$ multiplications of $n \times n$ h-circulant matrices by vectors for $h = f$ and $h = 1/e$, $2(3\tilde{r}+1)\tilde{r}$ additions/subtractions of n-dimensional vectors and $6\tilde{r}$ multiplications of such vectors by $e/(e-f)$, that is, to $O(\tilde{r}^2 n \log n)$ flops. Note that the computation of SVDs is not required in this case.

In [PBRZ99, Appendix A], it was proven that the output satisfies the bound (5.8). This bound immediately implies both correctness of the algorithm and convergence of $\tilde{X}_i$ to X with the rate $1 + b$ for a positive $b < 1$ provided that

$$\tilde{p}_0^{1-b}\mathrm{cond}_1(A)(1 + 0.5r||G||_1||H^T||_1||X||_1(1 + \tilde{p}_0^2)) \leq 1, \tag{5.9}$$

which extends (5.5) and Thm. 6.1. The comparison with (5.5) shows that a little stronger upper bound on the error norm of the initial approximation is needed now for the convergence of the refinement process, but if such a stronger bound is achieved, then the computation by Algorithm 2 is simpler since the displacement rank of Newton's iterates is controlled more directly, without involving the SVDs of the displacements.

6 Residual Correction for Matrix Inversion

6.1 General Residual Correction Method for Matrix Inversion

Newton's iteration is a special case (where $l = 2$) of the following more general residual correction algorithm: $X_{i+1,l} = (I + R_i + R_i^2 + \cdots + R_i^{l-1})X_{i,l}$, where $R_i = I + X_{i,l}A$. It can be easily deduced ([IK], p. 82) that $R_{i+1} = R_i^l$, and, therefore, $p_i = ||I + X_{i,l}A|| \leq p_{i-1}^l \leq p_0^{l^i}$. It is a tedious but straightforward exercise to extend our Algorithms 1 and 2 and their analysis to the case of $l > 2$ (see [IK] on some advantage of choosing $l = 3$ in the case of a general unstructured matrix A).

6.2 Residual Correction and Its Correlation to Newton's Iteration

The residual correction method can be restricted to the solution of a fixed linear system, rather than to computing the entire matrix inverse. If $X^{(i)}$ and $\mathbf{w}^{(i)}$ denote the current approximations to the matrix $X = -A^{-1}$ and to the solution $\mathbf{w} = A^{-1}\mathbf{b}$ to a linear system $A\mathbf{w} = \mathbf{b}$, respectively (computed in i recursive steps), then the next approximation,

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + X^{(i)}(A\mathbf{w}^{(i)} - \mathbf{b}) , \tag{6.1}$$

satisfies $\mathbf{w} - \mathbf{w}^{(i+1)} = (I + X^{(i)}A)(\mathbf{w} - \mathbf{w}^{(i)})$. Consequently, for any fixed operator norm (for matrices and vectors), we have

$$||\mathbf{w} - \mathbf{w}^{(i+1)}|| \leq p_i||\mathbf{w} - \mathbf{w}^{(i)}|| , \; p_i = ||I + X^{(i)}A)|| . \tag{6.2}$$

Actually, equation (6.1) is just a restriction of Newton's iteration (1.1). Indeed, post-multiply (1.1) by the vector $\mathbf{b}$, substitute $-\mathbf{w}^{(i+1)}$ for $X_{i+1}\mathbf{b}$ and $-\mathbf{w}^{(i)}$ for $X_i\mathbf{b}$, and obtain that $-\mathbf{w}^{(i+1)} = X_{i+1}\mathbf{b} = X_i\mathbf{b} + X_i(I + AX_i)\mathbf{b} = -\mathbf{w}^{(i)} + X_i(\mathbf{b} +$

$AX_i\mathbf{b}) = -\mathbf{w}^{(i)} + X_i(\mathbf{b} - A\mathbf{w}^{(i)})$. Therefore, $\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + X_i(A\mathbf{w}^{(i)} - \mathbf{b})$. It remains to substitute $X_i = X^{(i)}$ to obtain (6.1).

6.3 Application to a Toeplitz Linear System of Equations

We will next specify the latter variation of Newton's iteration in the case of a Toeplitz matrix, $A = T = [t_{i,j}]$, $t_{i,j} = t_{i-j}$ for all i and j. In this case, we will rely on two known formulae that express the inverse $-X = T^{-1}$ via the solution of two Toeplitz linear systems, $T\mathbf{y} = \mathbf{e}^{(0)}$ and $T\mathbf{x} = \mathbf{t}$, where $\mathbf{t} = [s,\ at_1 + bt_{1-n},\ at_2 + bt_{2-n},\ \ldots,\ at_{n-1} + bt_{-1}]^T$, for three fixed scalars, s, a and b. Each of these two expressions for the inverse relates the two equations of (6.1) for $\mathbf{b} = \mathbf{e}^{(0)}$ and $\mathbf{b} = \mathbf{t}$ to each other and to the matrix $-X = T^{-1}$.

In particular, to extend Algorithm 1 in this way, we will use the following known formula (cf. e.g. [BP93, p.136], and Remark 6.2 of this section):

$$-X \ = \ T^{-1} \ = \ L(\mathbf{x})L^T(ZJ\mathbf{y}) - L(\mathbf{y})L^T(ZJ\mathbf{x} - \mathbf{e}^{(0)}), \tag{6.3}$$

where $\mathbf{t} = [s, t_{1-n}, \ldots, t_{-1}]^T$, t_{i-j} denotes the (i,j)-th entry of T, s is any fixed scalar, say, $s = 0$ or $s = \max_{i,j} |t_{i-j}|$, J is the reflection matrix,

$$J = (j_{g,h})_{g,h=0}^{n-1},\ j_{g,h} = 1 \text{ if } g + h = n - 1,\ j_{g,h} = 0 \text{ otherwise},$$

so that $J[v_0, v_1, \ldots, v_{n-1}]^T = [v_{n-1}, \ldots, v_0]^T$ for any vector $[v_0, v_1, \ldots, v_{n-1}]^T$. Due to this expression and to (6.1), we may update the approximation $X^{(i)}$ to X as follows:

Algorithm 3, *Residual Correction for a Toeplitz Linear System.*

Input: A Toeplitz matrix T and the approximations $X^{(i)}$ to $X = -T^{-1}$, $\mathbf{x}^{(i)}$ to $\mathbf{x} = T^{-1}\mathbf{t}$, and $\mathbf{y}^{(i)}$ to $\mathbf{y} = T^{-1}\mathbf{e}^{(0)}$.

Output: New approximations $X^{(i+1)}$, $\mathbf{x}^{(i+1)}$ and $\mathbf{y}^{(i+1)}$.

Computations (cf. (6.1) for $A = T$ and (6.2)) : Compute and output

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + X^{(i)}(T\mathbf{x}^{(i)} - \mathbf{t}),$$

$$\mathbf{y}^{(i+1)} = \mathbf{y}^{(i)} + X^{(i)}(T\mathbf{y}^{(i)} - \mathbf{e}^{(0)}),$$

where we set

$$-X^{(i+1)} = L(\mathbf{x}^{(i+1)})L^T(ZJ\mathbf{y}^{(i+1)}) - L(\mathbf{y}^{(i+1)})L^T(ZJ\mathbf{x}^{(i+1)} - \mathbf{e}^{(0)}) . \qquad (6.4)$$

Algorithm 3 can be applied recursively for $i = 0, 1, \ldots$, provided that some initial approximations $\mathbf{x}^{(0)}, \mathbf{y}^{(0)}$ and $X^{(0)}$ to $\mathbf{x}, \mathbf{y}$ and X, respectively, are available. Furthermore, if just some initial approximations $\mathbf{x}^{(0)}$ and $\mathbf{y}^{(0)}$ are available, we may substitute $\mathbf{x} = \mathbf{x}^{(0)}$ and $\mathbf{y} = \mathbf{y}^{(0)}$ on the right-hand side of (6.3) and choose the resulting matrix as $-X^{(0)}$.

By using the operator ∇^f and equations (5.7), we may modify (6.3) and Algorithm 3 to replace the lower triangular Toeplitz matrices $L(\mathbf{x})$ and $L(\mathbf{y})$ by f-circulant matrices and the upper triangular Toeplitz matrices $L^T(ZJ\mathbf{y})$ and $L^T(ZJ\mathbf{x} - \mathbf{e}^{(0)})$ by $(1/e)$-circulant matrices. This can be also viewed as the specialization of Algorithm 2 to solving a Toeplitz linear system of equations. Namely, instead of (6.3), we will use the following result (cf. Remark 6.2 of this section).

Theorem 6.1. [GO92]. *Let* $T = (t_{i,j})_{i,j=0}^{n-1}$ *be a Toeplitz matrix,* $t_{i,j} = t_{i-j}$, $i, j = 0, 1, ..., n-1$. *Let* e, f *and* s *be three scalars,* $e \neq 0$, $ef \neq 1$. *Let two vectors* $\mathbf{x}$ *and* $\mathbf{y}$ *satisfy the linear systems of equations* $T\mathbf{y} = \mathbf{e}^{(0)}, T\mathbf{x} = \mathbf{t}(e, s)$, *where* $\mathbf{t}(e, s) = [s, t_1 - et_{1-n}, t_2 - et_{2-n}, ..., t_{n-1} - et_{-1}]^T$. *Then* T *is a non-singular matrix, and*

$$-X = T^{-1} = \frac{1}{1-ef}(Z_f(\mathbf{y})Z_{1/e}(\mathbf{x}) - Z_f(\mathbf{x} - (1-ef)\mathbf{e}^{(0)})Z_{1/e}(\mathbf{y})),$$

where $Z_g(\mathbf{v})$ *denotes the* g*-circulant matrix with the first column* $\mathbf{v}$ *(cf. Lemma 5.2).*

Now we will modify Algorithm 3 as follows:

(a) include the scalars e, f and s of Thm. 6.1 (say, write $f = -e = 1,\ s = 0$) into the input set of the algorithm,

(b) let $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(i+1)}$ approximate $T^{-1}\mathbf{t}(e,s)$, rather than $T^{-1}\mathbf{t}$, in particular, replace $\mathbf{t}$ by $\mathbf{t}(e,s)$ in the expression $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + X^{(i)}(T\mathbf{x}^{(i)} - \mathbf{t})$, and

(c) replace expression (6.4) for $-X^{(i+1)}$ by the following (based on Thm. 6.1):

$$-X^{(i+1)} =$$

$$\frac{1}{1-ef}(Z_f(\mathbf{y}^{(i+1)})Z_{1/e}(\mathbf{x}^{(i+1)}) - Z_f(\mathbf{x}^{(i+1)} - (1-ef)\mathbf{e}^{(0)})Z_{1/e}(\mathbf{y}^{(i+1)})). \qquad (6.5)$$

We will refer to this modification of Algorithm 3 as **Algorithm 4.** Since multiplication of a matrix $Z_f(\mathbf{v})$ by a vector is roughly twice as fast as multiplication of $L(\mathbf{v})$ or $L^T(\mathbf{v})$ by a vector, Algorithm 4 is roughly twice as fast as Algorithm 3.

Remark 6.1. *To use the equations (6.3)-(6.5) and Theorem 6.1 more efficiently, one may normalize the vector* $\mathbf{x}^{(0)}$ *by scaling the input matrix* T *or just change* e *and* f *so as to bring the norm* $||\mathbf{x}^{(0)}||$ *close to* $|1-ef| \cdot ||\mathbf{e}^{(0)}|| = |1-ef|$.

Remark 6.2. *Various modifications of the inverson formula (6.3) and, consequently, of Algorithm 3 are possible. In particular, one may express the matrices* $x_{0,0}X$ *and (if* $x_{0,0} \neq 0$*) then* $X = -T^{-1} = (x_{i,j})$*, via the columns* $\mathbf{y} = X\mathbf{e}^{(0)}$ *and* $\mathbf{x} = X\mathbf{e}^{(n-1)}$ *based on the Gohberg-Semencul celebrated formula (cf. e.g. [BP93, p.135], and [GO92]). We choose (6.3) rather than the latter formula to avoid the division by* $x_{0,0}$*, since such a divisor may vanish or may have a too small magnitude even for a well-conditioned input matrices* T*. Similar comments apply to the inversion formulae of Thm. 6.1 and Algorithm 4 of this section. The expressions of Thm. 6.1 is closely related to (5.7) in the case of a non-singular Toeplitz matrix* A*.*

6.4 Estimates for The Convergence Rate

Since Algorithms 3 and 4 are the specializations of Algorithms 1 and 2, the estimates for the convergence rates of Algorithms 1 and 2 can be extended to Algorithms 3 and 4. Next, however, we will estimate the convergence rate of Algorithms 3 and 4 directly. Let us write

$$e_i = \max\{||\mathbf{x}^{(i)} - \mathbf{x}||_1/||\mathbf{x}||_1,\ ||\mathbf{y}^{(i)} - \mathbf{y}||_1/||\mathbf{y}||_1\}.$$

By combining the latter definition and the norm bounds (6.2) of section 6.2 for $\mathbf{w} = \mathbf{x}$ and $\mathbf{w} = \mathbf{y}$, we immediately deduce that

$$||\mathbf{x}^{(i+1)} - \mathbf{x}||_1 \ \le\ p_i||\mathbf{x}^{(i)} - \mathbf{x}||_1 \ \le\ p_i e_i||\mathbf{x}||_1\ , \tag{6.6}$$

$$||\mathbf{y}^{(i+1)} - \mathbf{y}||_1 \ \le\ p_i||\mathbf{y}^{(i)} - \mathbf{y}||_1 \ \le\ p_i e_i||\mathbf{y}||_1\ , \tag{6.7}$$

and consequently,

$$||\mathbf{x}^{(i+1)}||_1 \ \le\ (1 + p_i e_i)||\mathbf{x}||_1\ ,\ ||\mathbf{y}^{(i+1)}||_1 \ \le\ (1 + p_i e_i)||\mathbf{y}||_1\ . \tag{6.8}$$

In parts B and C of the appendix we will prove the following estimates.

Lemma 6.1. *(a)* $E_{i+1} = ||X - X^{(i+1)}||_1 \le p_i e_i||\mathbf{y}||_1(2(n-1)(2 + p_i e_i)||\mathbf{x}||_1 + 1)$, *if Algorithm 3 is applied.*
(b) $E_{i+1} = ||X - X^{(i+1)}||_1 \le p_i e_i||\mathbf{y}||_1(||\mathbf{x}||_1(1 + p_i e_i) + 1)$, *if Algorithm 4 is applied and if* $f = -e = 1$ *in (6.5) and in Theorem 6.1.*

We also have the bound $p_{i+1} \le ||A||_1 E_{i+1}$, and by (6.2), $e_{i+1} \le p_i e_i$. By applying Lemma 6.1, we obtain that

$$p_{i+1} \ \le\ p_i e_i||A||_1||\mathbf{y}||_1(2(n-1)(2 + p_i e_i)||\mathbf{x}||_1 + 1)$$

and

$$p_{i+1} \ \le\ p_i e_i||A||_1||\mathbf{y}||_1(||\mathbf{x}||_1(1 + p_i e_i) + 1)\ ,$$

where Algorithms 3 and 4 are applied, respectively. Let us write

$$\nu = ||\mathbf{y}||_1(2(n-1)(2+p_0e_0)||\mathbf{x}||_1+1) ,$$

if Algorithm 3 is applied, and

$$\nu = ||\mathbf{y}||_1(||\mathbf{x}||_1(1+p_0e_0)+1) ,$$

if Algorithm 4 is applied. In both cases, we have $p_{i+1} \leq p_ie_i||A||_1\nu$, provided that $p_ie_i \leq p_0e_0$, for all i.

Now suppose that p_0 and e_0 are sufficiently small such that

$$p_0 < \rho \ , \ e_0||A||_1\nu < \rho \tag{6.9}$$

for a fixed $\rho < 1$. Then we obtain that $p_1 \leq p_0e_0||A||_1\nu < p_0\rho < \rho^2$, so that $p_1e_1 < \rho^2p_0e_0 < p_0e_0$. Recursively, we obtain by induction that

$$p_{i+1} \leq p_ie_i||A||_1\nu \leq p_i(e_i/e_0)e_0||A||_1\nu \leq (e_i/e_0)p_i\rho < \rho^{2^{i+1}},$$

$$e_{i+1} \leq p_ie_i < \rho^{2^{i+1}-1}e_0, \ i = 0,1,\ldots .$$

This implies quadratic convergence with the base ρ assuming the initial upper bounds (6.9) on e_0 and p_0, which extend the respective bounds of (5.5) and (5.8) to the Toeplitz case.

7 Homotopic Residual Correction (HRC)

7.1 An HRC Algorithm for a Positive Definite Matrix

A reliable solution of the initialization problem for the RC processes is given by *homotopic RC processes*, to be referred to as *HRC processes* and studied next.

Algorithm 7.1. A homotopic RC process for a positive definite matrix.

Input: *an $n \times n$ Hermitian positive definite matrix, a non-negative ε, a positive λ_1^+ such that* spectrum$(M) = \{\lambda_1, \dots, \lambda_n\}$, *where*

$$\lambda_1^+ \geq \lambda_1 = \|M\|_2 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq \lambda_n^- > 0, \tag{7.1}$$

and a black box RC process (2.1) (scaled or unscaled and with any selected levels of compression in the case of a structured input).

Initialization: *Fix some values θ_h, $0 < \theta_h < 1$, $h = 0, 1, \dots$, and write (cf. (1.4), (1.5))*

$$M_0 = M + t_0 I, \ t_0 = \lambda_1^+/\theta_0, \ X_0 = t_0^{-1} I, \tag{7.2}$$

$$M_{h+1} = t_{h+1} I + M = M_h - \Delta_h I, \ \ \Delta_h = t_h - t_{h+1} > 0, \ \ h = 0, 1, \dots. \tag{7.3}$$

Apply the selected black box RC process (2.1) for $M = M_0$ and X_0 of (7.2) to approximate M^{-1} by $\tilde{X}_0$ such that

$$\|R(\tilde{X}_0, M_0^{-1})\| \leq \varepsilon. \tag{7.4}$$

Computations: *Stage h, $h = 0, 1, \dots$. Compute an upper bound η_h on the norm*

$$\|M_h^{-1}\|_2 = \frac{1}{t_h + \lambda_n} \tag{7.5}$$

(see Remark 7.1). Compute

$$\Delta_h = \theta_h/\eta_h. \tag{7.6}$$

Apply the black box RC process with $X_0 = \tilde{X}_h$, *replacing* M *by* M_{h+1} *if* $t_{h+1} > 0$. *If* $t_{h+1} \geq 0$, *write* $H = h+1$, *use* M *instead of* M_{h+1}, *and compute a matrix* $\tilde{X}_{h+1}$ *such that*

$$\|R(M_{h+1}, \tilde{X}_{h+1})\| \leq \varepsilon. \tag{7.7}$$

Output: $\tilde{X}_H$ *approximating* M^{-1} *and stop.*

The algorithm is completely defined as soon as we fix an RC process (2.1) (including its stopping criterion and, for structured matrices M, the policy of the compression of the displacements), although we may modify the algorithm to allow this policy and the residual norm bound ε vary with h.

Let us show correctness of the algorithm for $\varepsilon = 0$. First observe that for the residual R_0 of (2.2), we have

$$R_0 = R(M_0, t_0^{-1}I) = I - t_0^{-1}M_0 = -t_0^{-1}M,\ r_0 = \|R(M_0, t_0^{-1}I)\|_2 \leq \theta_0. \tag{7.8}$$

Further, deduce from (5.3) that

$$R(M_{h+1}, M_h^{-1}) = \Delta_h M_h^{-1},$$
$$r_{h+1} = \|R(M_{h+1}, M_h^{-1})\|_2 = \Delta_h \|M_h^{-1}\|_2 = \Delta_h/(t_h + \lambda_n). \tag{7.9}$$

Write

$$\lambda_{n,h} = 1/\eta_h - t_h \leq 1/||M_h^{-1}||_2 - t_h = \lambda_n \tag{7.10}$$

and observe that the value $\lambda_n^- = \lambda_{n,h}$ satisfies bound (7.1). Finally, (7.6) implies that

$$r_{h+1} \leq \theta_h \text{ for all } h. \tag{7.11}$$

In the next two sections, we estimate the overall numbers of the RC steps required for the inversion of a general unstructured Hermitian positive definite matrix M and optimize this number by choosing appropriate bounds θ_h for a

fixed order of convergence q of the basic RC process. In Section 8, we extend the algorithm to the case of a general indefinite matrix M. In Section 9–11, we cover extensions to the cases where the matrix M is structured and compression of the displacements is applied, where the matrix M is singular, and/or where a numerical inverse of M is computed.

Remark 7.1. *We have* $\|M_h^{-1}\|_1/\sqrt{n} \le \|M_h^{-1}\|_2 \le \|M_h^{-1}\|_1$ *for an* $n \times n$ *Hermitian matrix* M_h^{-1}. *Sharper upper bound* η_h *on the matrix norm (7.5) can be obtained by applying the power or Lanczos methods [GL96]. If an estimate* η_h *is sufficiently sharp for a fixed* $h = k$ *(say for* $h = 1$*), a close upper bounds* η_{k+i} *can be computed based on the following simple expression:*

$$\eta_{k+i} = \frac{1}{t_{k+i} + \lambda_{n,k}}, \quad \lambda_{n,k} = 1/\eta_k - t_k, \quad i = 1, 2, \ldots$$

(see (7.3)-(7.11)).

Remark 7.2. *The homotopic process of (7.2), (7.3) has trajectory* $M(t) = M + tI$ *which for* $t > 0$ *is better conditioned than the input matrix* M. *That is, one may easily verify that*

$$\kappa(M(t)) \le \kappa(M) \text{ for } t \le 0. \tag{7.12}$$

The same inequality can be easily verified for the modification of the homotopic process in Section 8 in the indefinite Hermitian case.

Remark 7.3. *The approach allows variations. For instance, instead of process (7.2), (7.3), we may apply homotopic process (1.3) or the dual process*

$$M_{h+1} = I + t_{h+1}M = M_h + (t_{h+1} - t_h)M, \quad h = 0, 1, \ldots,$$

followed at the end by a single step (7.3) or a few steps (7.3). The resulting computations can be analyzed similarly to process (7.3).

7.2 The Number of Homotopic Steps

To simplify our subsequent analysis, we next assume that the values $\lambda_{n,h}$ and θ_h are invariant in h, that is, $\lambda_{n,h} = \lambda_n^-$ for all $h \geq 1$ (cf. (7.10) and Remark 7.1). Then by virtue of (7.3), (7.5), (7.6), and (7.10), we have

$$t_{h+1} + \lambda_n^- = (1 - \theta_h)(t_h + \lambda_n^-), \quad h = 0, 1, \ldots, H-1.$$

Therefore,

$$t_{h+1} + \lambda_n^- = (t_0 + \lambda_n^-) \prod_{i=0}^{h} (1 - \theta_i), \quad h = 0, 1, \ldots, H-1,$$

$$t_H \leq 0 \text{ if } \lambda_n^- \geq (t_0 + \lambda_n^-) \prod_{h=0}^{H-1} (1 - \theta_h).$$

Furthermore, let θ_h be invariant in h, that is, let $\theta_h = \theta$ for all h. Substitute $t_0 = \lambda_1^+/\theta$ of (7.2) and rewrite the latter inequality as follows:

$$\frac{1}{(1-\theta)^H} \geq \lambda_1^+/(\theta\lambda_n^-) + 1,$$

$$H \geq -\log(1 + \lambda_1^+/(\theta\lambda_n^-))/\log(1-\theta).$$

Choose the minimum integer H satisfying this bound, that is,

$$H = \left\lceil \frac{\log(1 + \lambda_1^+/(\theta\lambda_n^-))}{\log(1/(1-\theta))} \right\rceil \tag{7.13}$$

homotopic steps are sufficient. Substitute

$$\theta = K/(1+K) \tag{7.14}$$

and rewrite (7.13) as follows:

$$H = \left\lceil \frac{\log(1 + (K+1)\lambda_1^+/(K\lambda_n^-))}{\log(1+K)} \right\rceil. \tag{7.15}$$

7.3 The Overall Number of RC Steps

At each homotopic step, the number of RC steps depends on the bound θ on the initial residual norm (to be assumed invariant at all homotopic steps), the order q of convergence of the selected RC process, and the stopping criterion for this process. We assume some fixed order q for each process (2.1) given a general unstructured matrix M and scalars p and c_{i+1}, $i = 0, 1, \ldots$. In particular, $q = p$ for unscaled processes (2.1), (2.3).

7.4 Critical and Refinement Stages of an RC Process

Estimating the number of RC steps at the i-th homotopic step, we treat separately its initial *critical stage*, where the residual norm decreases below $1/e = 1/2.718281\ldots = 0.367819\ldots$, and the subsequent *refinement stage*, where the residual norm decreases below a fixed target bound ν_i for the output approximation X_j to M_i^{-1} (compare a similar partition of a non-homotopic process in Section 2). We write $\nu_H = \epsilon$ and $\nu_i = \nu$ for all $i < H$, and choose the scalar $\nu = \nu(\theta)$ sufficiently small to ensure that the computed approximations are close enough to the matrices M_i^{-1} to serve as initial approximations at the next homotopic steps.

7.5 The Number of RC Steps at The Refinement Stages

Processes (2.1) with the order of convergence q decrease the residual norm from $1/e$ to e^{-q^g} in g RC steps (cf. (2.4)). Therefore, at the H-th homotopic step, the refinement requires

$$\gamma = \lceil (\log \ln(1/\epsilon))/\log q \rceil \tag{7.16}$$

RC steps, whereas

$$\beta = \lceil (\log \ln(1/\nu))/\log q \rceil \tag{7.17}$$

refinement steps are sufficient for the transition from $1/e$ to ν for each $i < H$.

Summarizing, we have a total of at most

$$P = \gamma + (H-1)\beta \tag{7.18}$$

RC steps at the refinement stages of all homotopic steps of the HRC algorithm. Bound (7.16) applies to the number of all refinement RC steps of the non-homotopic processes of Section 2 (for the same q and ϵ). Bound (7.2) covers the $(H-1)\beta$ refinement RC steps particular to the HRC processes. Practically, β is quite small. For instance, for $q = 4$, the bound e^{-16} is achieved in two steps. The specific choice of the bound ν can be guided by the following simple estimate.

Proposition 7.1. *Let*

$$\|I - XM_{h-1}\| \leq \nu, \tag{7.19}$$

$$\|I - M_{h-1}^{-1}M_h\| \leq \theta_h \tag{7.20}$$

for any fixed matrix norm. Then

$$\|I - XM_h\| \leq (1+\nu)\theta_h + \nu.$$

Proof.

$$\begin{aligned} \|I - XM_h\| &\leq \nu + \|XM_{h-1} + XM_h\| \\ &\leq \nu + \|XM_{h-1}\|\ \|I - M_{h-1}^{-1}M_h\| \\ &\leq \nu + (1+\nu)\theta_h. \end{aligned}$$

□

7.6 The Nnumber of RC Steps at The Critical Stages

Let α denote the number of RC steps used at the critical stage of a homotopic step. Then we have

$$\frac{1}{\theta_h^{q^\alpha}} = (1+\frac{1}{K})^{q^\alpha} \approx e,\ q^\alpha \approx \frac{1}{\ln(1+\frac{1}{K})} \approx K,\ \alpha \approx \frac{\log K}{\log q} \tag{7.21}$$

provided that θ is close to 1, that is, that K is large.

By combining (7.15) and (7.21) for $\theta_h = \theta$ for all h, we estimate the overall number of RC steps at all critical stages of the entire HRC process:

$$N = \alpha H \approx \frac{\log(\lambda_1^+/\lambda_n^-)\log K}{\log(K+1)\log q} \leq N^+ = \frac{\log(\lambda_1^+/\lambda_n^-))}{\log q}. \tag{7.22}$$

7.7 The Overall Number of RC Steps in Homotopic and Non-Homotopic Processes

Based on (7.15), (7.17)–(7.20), and (7.22), one may immediately estimate the overall number

$$N + P = \alpha H + \gamma + (H-1)\beta$$

of the RC steps of the entire HRC algorithm. This is the same bound as in Section 2 for non-homotopic RC processes both with scaling (for q=4) and without it (for q=2).

8 Inversion of Indefinite Matrices

We may extend our HRC algorithm of the previous section to compute numerically the inverse M^{-1} of any non-singular matrix based on the equations

$$M^{-1} = M^*(MM^*)^{-1} = (M^*M)^{-1}M^* \tag{8.1}$$

because the matrices MM^* and M^*M are Hermitian (or real symmetric) and positive definite. This standard symmetrization, however, has the well-known price of squaring the condition number and, consequently, of a substantial slowdown of the HRC algorithm (cf. (7.22)). Let us show a simple remedy in the case where M is a non-singular Hermitian (or a real symmetric) indefinite matrix M. Recall that the inversion of any non-singular input matrix M reduces to the inversion of the Hermitian or real symmetric matrix

$$N = \begin{pmatrix} 0 & M \\ M^* & 0 \end{pmatrix}, \tag{8.2}$$

where

$$N^{-1} = \begin{pmatrix} 0 & (M^*)^{-1} \\ M^{-1} & 0 \end{pmatrix}, \quad \kappa(N) = \kappa(M).$$

Let λ^- and λ^+ be two fixed positive values such that

$$\lambda^- \le |\lambda| \le \lambda^+$$

for every eigenvalue λ of M. Then for any fixed sequence of real θ_h, $0 < \theta_h < 1$, $h = 0, 1, \ldots$, we define an HRC process by (7.2)–(7.7), for η_h still denoting an upper bound on the norm $||M_h^{-1}||_2$ but with the matrix I replaced by the matrix $I\sqrt{-1}$. That is, our HRC algorithm (which can be applied to any Hermitian input matrix M) is now defined by the equations

$$M_0 = M + t_0 I\sqrt{-1}, \quad t_0 = \lambda^+/\theta_0, \tag{8.3}$$

$$X_0 = -t_0^{-1} I\sqrt{-1} \tag{8.4}$$

(replacing (7.2)), and

$$M_{h+1} = t_{h+1}I\sqrt{-1}+M = M_h-\Delta_h I\sqrt{-1},\ \Delta_h = t_h-t_{h+1} > 0,\ h = 0,1,\dots \quad (8.5)$$

(replacing (7.3)). (8.3)–(8.5) immediately imply bounds (7.8) and (7.11) for $\eta_h \geq ||M_h^{-1}||_2$ and Δ_h of (7.6).

Let us extend our analysis presented in Section 7. First note that the equation

$$||M_h^{-1}||_2 = ((t_h^2 + (\lambda^-)^2)^{-1/2} \text{ for all } h \quad (8.6)$$

replaces (7.5). Then again let us simplify the analysis, similarly to Section 7. Assume that $\eta_h = (t_h^2+(\lambda^-)^2)^{-1/2}$ (cf. Remark 7.1) and $\theta_h = \theta$ for all h. It follows that

$$t_{h+1} = t_h - \Delta_h = t_h - (t_h^2 + (\lambda^-)^2)^{1/2}\theta < t_h - \theta\max\{t_h, \lambda^-\},\ \ h = 0,1,\dots.$$

Therefore, $t_{h+1} < 0$ where $(1-\theta)^h t_0 \leq \theta\lambda^-$. Substitute $t_0 = \lambda^+/\theta$ and obtain that $t_H \leq 0$ where

$$H - 1 = \left\lceil \frac{\log(\lambda^+/(\theta^2\lambda^-))}{\log(1/(1-\theta))} \right\rceil.$$

The latter bound is within the term $\eta = 1 + \lceil(\log(1/\theta))/\log(1/(1-\theta))\rceil$ from bound (7.13) for $\lambda_1^+ = \lambda^+$ and $\lambda_n^- = \lambda^-$. This term is at most 2 for $\theta \geq 1/2$. On the other hand, our estimates of Section 7.3 for the numbers of critical and refinement steps performed in each homotopic step remain unchanged (these estimates are completely defined by the parameters ϵ, ν, and θ). Therefore, up to replacing λ_n^- by λ^- and λ_1^+ by λ^+ and performing at most $a = \eta\lceil\log_q((\log\nu)/\log\theta)\rceil$ additional RC steps, the estimates of Section 7 apply to the Hermitian indefinite case as well. The latter bound a is relatively small, and we ignore it in Table 8.1, which summarizes our estimates for the overall numbers of RC steps in the HRC processes and non-homotopic RC processes applied to the same general Hermitian matrix M. (Table 8.1 uses γ of (7.1), H of (7.13), (7.15), and $\kappa_+(M)$ equal to either

λ_1^+/λ_n^- or λ^+/λ^-.) According to these estimates, the HRC processes use roughly as many RC steps as non-homotopic RC processes for the inversion of a Hermitian positive definite input matrix M where M is positive definite and roughly by twice fewer critical RC steps and as many refinement RC steps where M is indefinite.

Table 8.1: Numbers of RC steps required for numerical inversion of Hermitian matrices M.

	RC Processes	HRC Processes
indefinite M	$\log_2 \kappa_+(M) + \gamma + O(1)$	$0.5 \log_2 \kappa_+(M) + \gamma + O(H)$
positive definite M	$0.5 \log_2 \kappa_+(M) + \gamma + O(1)$	$0.5 \log_2 \kappa_+(M) + \gamma + O(H)$

9 RC and HRC Processes with Compression

Suppose an RC process with compression has been applied to a structured input matrix M. Then compression of the displacements perturbs the computed approximations to the inverse, and this may destroy convergence, particularly at the critical RC steps, at which the convergence is more fragile. A natural recipe is to use no compression or limited compression until close approximations X_i to M^{-1} are computed. (Recall Remark 3.2.) How close should these approximations be?

Theorems 3.4 and 3.5 show the level of approximation starting at which rapid convergence is guaranteed for the Newton–Toeplitz Iteration (6.5) and for the unscaled Newton-Structured RC process (2.8), (2.3), even under the *maximal compression*, such that the number of the untruncated singular values of the displacements of the computed approximations is set to be equal to the displacement rank of M. On the other hand, the techniques of Section 2 (cf. (2.6) and (2.7)) fall short of even approaching this level. If we start with an initial approximation obtained according to the recipes of Section 2, then to ensure the desired levels of Theorems 3.4 and 3.5 using no compression, we should allow an increase of the displacement rank to n, which means complete loss of the matrix structure. In this case already a single RC step would become too expensive in terms of the number of flops involved.

Practically, the non-homotopic structured RC processes are frequently effective, however. That is, according to the experiments reported in our Appendix B, [P01a], and [BM,a], the initial approximation policies of Section 2 under the maximal compression or under compression close to the maximal frequently enable sufficiently rapid convergence in the Toeplitz case. Furthermore, in a large portion of test runs of non-homotopic processes (2.3), (3.7) for $p = 2$ and Toeplitz input matrices under Approach I, the residual 2-norm grew above 1 and sometimes well

above 1 in the initial RC steps, and then iteration still converged. Since every RC step (2.3), (3.3) for $p = 2$ squares the 2-norm of the residual, the only explanation of this phenomenon is that the compression frequently decreases the residual norm, thus bringing the approximation *closer* to the inverse, so that the estimates of Theorems 3.4 and 3.5 are overly pessimistic.

HRC processes with compression is an alternative approach supported both experimentally (see Appendix B) and theoretically [P92]. It is proved in [P92] that $O((n \log^3 n) \log \kappa_+(M) + (n \log n) \log\log(1/\epsilon))$ flops are sufficient to approximate M^{-1} for an $n \times n$ Toeplitz-like matrix M. The latter bound is supported in [P92] by an HRC algorithm with the maximal compression (to the level of the displacement rank of M) throughout the computations, and the convergence is controlled via the choice of the sizes of the homotopic steps. Further progress could be achieved based on simultaneous optimization of two groups of parameters, that is, the tolerance values θ_h defining the step sizes Δ_h and the levels of compression based on experimental computations.

HRC processes could be further improved for specific structures of the input matrices. For instance, for real non-singular Toeplitz matrices T, one may achieve symmetrization without doubling the matrix size, simply in the transition to the Hankel matrices JT or TJ, which are real symmetric and satisfy the equations $T^{-1} = (JT)^{-1}J = J(TJ)^{-1}$.

On the other hand, the structure of Cauchy or Vandermonde types is not generally preserved in the transition from a matrix M to the matrices M_0 of (7.2) and (8.4). The problem is solved in the next section where we extend the HRC processes to the case where M is a Hermitian matrix and $M_0 = \hat{M}$ or $M_0 = \hat{M}\sqrt{-1}$ for any Hermitian and positive definite matrix $\hat{M}$.

Example 9.1. *Pick matrices*

$$M = P = \left(\frac{\mathbf{u}_i^* \mathbf{v}_k}{z_i + z_k^*} \right)_{i,k=1}^{n}$$

where $\mathbf{u}_i$ *and* $\mathbf{v}_k$ *are vectors of a fixed dimension* d; z_i *are scalars,* $Im\, z_i > 0$, *and* z_k^* *are the complex conjugates of* z_k *for all* i *and* k. *Pick matrices define the Nevanlinna-Pick celebrated problem of rational interpolation [BGR90] and the matrix Nehari problem of rational approximation [BGR90a], [GO94b], [OP98]. The problem is solvable if and only if the Pick matrix is positive definite. One may apply our HRC processes, but the Cauchy structure of the Pick matrices* $M = P$ *is not preserved in the transition to the matrices* M_0 *of (7.2) and (8.4). The structure is much better preserved, however, if we choose*

$$M_h = M + t_h M_0, \quad M_0 = \left(\frac{1}{z_i + z_k^*} \right)_{i,k=1}^{n}$$

or, more generally,

$$M_0 = \left(\frac{\mathbf{x}_i^* \mathbf{y}_k}{z_i + z_k^*} \right)_{i,k=1}^{n}$$

where $\mathbf{x}_i$ *and* $\mathbf{y}_k$ *are* l*-dimensional column vectors for a fixed small non-negative integer* l. *Our extension of the HRC processes in the next section covers the above initialization proposed in the case of Pick matrices.*

10 An HRC Process with Generalized Initialization

Motivated by the applications to the inversion of structured matrices, let us extend homotopic processes and their analysis by allowing more general choice of the initial matrix M_0.

First assume that M and M_0 is any fixed pair of positive definite matrices, where M_0 is readily invertible, spectrum$(M_0) = \{\mu_1, \dots, \mu_n\}$,

$$\mu_1^+ \geq \mu_1 \geq \mu_2 \geq \cdots \geq \mu_n \geq \mu_n^- > 0, \tag{10.1}$$

and the values μ_1^+ and μ_n^- are available. Now recursively define scalars $t_1, \dots, t_{H-1}$ and matrices

$$M_{h+1} = t_{h+1}M_0 + M = M_h + (t_{h+1} - t_h)M_0, \quad h = 0, 1, \dots, H-1, \tag{10.2}$$

where $t_1 > t_2 > \cdots > t_{H-1} > t_H = 0$.

One may rewrite (10.2) as $M_{h+1} = M_0(t_h I + M_0^{-1}M)$ and apply our previous study to the inversion of the matrix $M_0^{-1}M$, but we avoid shifting to this matrix directly. We deduce that

$$||I - (t_1 M_0)^{-1}M_1||_2 \leq ||M_0^{-1}M/t_1||_2 \leq ||M_0^{-1}||_2||M||_2/t_1 \leq \lambda_1^+/(t_1\mu_n^-),$$

for λ_1^+ of (5.1) and choose

$$t_1 = \lambda_1^+/(\theta_0\mu_n^-) \tag{10.3}$$

so that $||I - (t_1 M_0)^{-1}M_1||_2 \leq \theta_0$. Invert M_1 by applying processes (2.1) for $X_0 = t_1 M_0$.

Now deduce from (10.2) that

$$I - M_h^{-1}M_{h+1} = (t_h - t_{h+1})M_h^{-1}M_0,$$
$$||I - M_h^{-1}M_{h+1}||_2 \leq (t_h - t_{h+1})||M_h^{-1}||_2||M_0||_2. \tag{10.4}$$

Substitute the bound

$$||M_0||_2 \leq \mu_1^+$$

and obtain that $||I - M_h^{-1}M_{h+1}||_2 \leq \theta_h$ if $(t_h - t_{h+1})\mu_1^+||M_h^{-1}||_2 \leq \theta_h$ or, equivalently, if $t_{h+1} \geq t_h - \theta_h/(\mu_1^+||M_h^{-1}||_2)$. Recall that, clearly,

$$||M_h^{-1}||_2 \leq 1/(t_h\mu_n^- + \lambda_n^-)$$

for all h and for λ_n^- of (7.1) [Par80, p.191], write

$$t_{h+1} = t_h - (t_h\mu_n^- + \lambda_n^-)\theta_h/\mu_1^+, \tag{10.5}$$

and deduce (5.11). Now, invert the matrices M_{h+1} by applying processes (2.1) for $X_0 = M_h^{-1}$ and for $h = 1, 2, \ldots, H-2$, until the value t_{h+1} of (10.5) becomes non-positive for $h = H-1$. Then at the last homotopic step, invert M instead of M_H.

Clearly, the estimates of Section 7 for the number of RC steps at each homotopic step apply to the above generalized HRC process as well.

Let us next estimate the number of homotopic steps H, in terms of the parameters t_1, θ_h, $\kappa^+ = \mu_1^+/\mu_n^-$, the lower bounds λ_n^- and μ_n^- on the eigenvalues of the matrices M and M_0. Substitute the expression $\kappa^+ = \mu_1^+/\mu_n^-$ into (10.5) for $h = 0, 1, \ldots, H-1$ and obtain that

$$\begin{aligned} t_{h+1} &= t_h(1 - \theta_h/\kappa^+) - \theta_h\lambda_n^-/\mu_1^+ \\ t_{h+1} + \kappa^+\lambda_n^-/\mu_n^- &= (t_h + \kappa^+\lambda_n^-/\mu_1^+)(1 - \theta_h/\kappa^+) \\ &= (t_1 + \kappa^+\lambda_n^-/\mu_n^-)\prod_{i=0}^{h}(1 - \theta_i/\kappa^+). \end{aligned} \tag{10.6}$$

Therefore, we have $t_{h+1} \leq 0$ if

$$(t_1 + \kappa^+\lambda_n^-/\mu_n^-)\prod_{i=0}^{h}(1 - \theta_i/\kappa^+) \geq \kappa^+\lambda_n^-/\mu_n^-,$$

that is, if

$$1 + t_1\mu_n^-/(\lambda_n^-\kappa^+) \geq 1/\prod_{i=0}^{h}(1-\theta_i/\kappa^+).$$

Assuming that $\theta_h = \theta$ is invariant in h, we arrive at $t_H \leq 0$ for

$$H = 1 + \left\lceil \frac{(\log(1 + t_1\mu_n^-/(\lambda_n^-\kappa^+)))}{(\log(1-\theta/\kappa^+)^{-1})} \right\rceil \tag{10.7}$$

and t_1 of (10.3).

Finally, if M is any non-singular matrix, we may apply symmetrization recipes (8.1) or (8.2) to extend our algorithm of this section. In particular, recipe (8.2) reduces the problem to the case where M is a Hermitian (or real symmetric) but not necessarily positive definite matrix. Then we may extend HRC process (10.2)–(10.5) where we keep equations (10.2)–(10.3), choose the matrix M_0 equal to $\hat{M}\sqrt{-1}$ for a fixed positive definite matrix $\hat{M}$, and modify (10.4)–(10.5) to ensure that $||I - M_h^{-1}M_{h+1}||_2 \leq \theta_h$ for all h.

Let us complete the description of this extended homotopic process. Assume that bounds (10.1) still hold where $\{\mu_1 \ldots, \mu_n\} = \text{spectrum}(M)$ and each eigenvalue λ of the input matrix M satisfies the bounds

$$0 < \lambda^- \leq |\lambda| \leq \lambda^+ \tag{10.8}$$

for two fixed positive values λ^- and λ^+. Now write

$$t_{h+1} = t_h - (\theta_h/\mu_1^+)((\lambda^-/\kappa^+)^2 + (t_h\mu_n^-)^2)^{1/2}, \quad \kappa^+ = \mu_1^+/\mu_n^-, \tag{10.9}$$

$h = 0, 1, \ldots, H-1$.

Let us deduce bounds (5.11). Recall the following well-known theorem [Par80, proof of Theorem 15-3-3].

Theorem 10.1. *Let M and $\hat{M}$ be two Hermitian matrices. Let the matrix $\hat{M}$ be positive definite, such that*

$$\hat{M} = U\Sigma^2U^* \tag{10.10}$$

for a unitary matrix U, $U^*U = UU^* = I_n$, *and a diagonal matrix*

$$\Sigma = \mathrm{diag}(\sigma_i)_{i=1}^n,\ \mu_1^+ \geq \sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_n^2 \geq \mu_n^- > 0.$$

Then there exists a unitary matrix V, $V^*V = VV^* = I_n$, *such that*

$$D = V^*\Sigma^{-1}U^*MU\Sigma^{-1}V \tag{10.11}$$

is a real diagonal matrix.

Corollary 10.1. *Under the notation of (10.1), (10.8), and Theorem 6.1, we have*

$$\|M_h^{-1}\|_2^2 \leq (\mu_n^-)^{-2}((\lambda^-/\mu_n^+)^2 + t_h^2)^{-1} = ((\lambda^-/\kappa^+)^2 + (t_h\mu_n^-)^2)^{-1}$$

for $h = 1, 2, \ldots$ *where* $\kappa^+ = \mu_1^+/\mu_n^-$.

Proof. By combining (10.10) and (10.11), obtain that

$$M_h = M + t_h\sqrt{-1}\hat{M} = U\Sigma V(D + t_hI\sqrt{-1})V^*\Sigma U^*,$$

$$M_h^{-1} = U\Sigma^{-1}V(D + t_hI\sqrt{-1})^{-1}V^*\Sigma^{-1}U^*.$$

Therefore,

$$\|M_h^{-1}\|_2 \leq \|\Sigma^{-2}\|_2\ \|(D + t_hI\sqrt{-1})^{-1}\|_2 \leq \left(\frac{1}{\mu_n^-}\right)^2 \left(\frac{1}{\|D^{-1}\|_2^2} + t_h^2\right)^{-0.5}.$$

On the other hand, we deduce from (10.1), (10.8), and (10.11) that

$$\|D^{-1}\|_2 \leq \|\Sigma^2\|_2\ \|M^{-1}\|_2 \leq \mu_1^+/\lambda^-.$$

Substitute the latter bound into our estimate for the norm $\|M_h^{-1}\|_2$ and obtain that

$$\|M_h^{-1}\|_2^2 \leq (\mu_n^-)^{-2}((\lambda^-/\mu_1^+)^2 + t_h^2)^{-1} = ((\lambda^-/\kappa^+)^2 + (t_h\mu_n^-)^2)^{-1}.$$

□

Relations (10.1), (10.2), (10.4), (10.9), and Corollary 10.1 together immediately imply (5.11). Let us compare the estimate of Corollary 10.1 and the bound $\|M_h^{-1}\|_2 \le 1/(t_h\mu_n^- + \lambda_n^-)$. The two estimates are close to one another provided that the terms λ_n^- and λ^-/κ^+ are dominated by the term $t_h\mu_n^-$. If the term λ^-/κ^+ dominates, the bound of Corollary 10.1 may be larger by roughly the factor of $\kappa^+\lambda_n^-/\lambda^-$.

(10.9) implies the crude bounds

$$t_{h+1} \le t_h - (\theta_h/\mu_1^+)(\lambda^-/\kappa^+ + t_h\mu_n^-),\ h = 1, 2, \ldots.$$

Consequently,

$$t_{h+1} + \lambda^-/\mu_1^+ \le (1 - \theta_h/\kappa^+)(t_h + \lambda^-/\mu_1^+) \le \ldots \le (t_1 + \lambda^-/\mu_1^+)\prod_{i=1}^{h}(1 - \theta_i/\kappa^+).$$

The latter inequality implies that the value t_H is non-positive for

$$H \le 1 + \lceil(\log(1 + t_1\mu_1^+/\lambda^-))/\log(1 - \theta/\kappa^+)^{-1}\rceil$$

provided that $\theta_h = \theta$ for all h.

11 Extensions and Generalizations

It is well known and easily verified that the unscaled RC processes (2.1), (2.3) and the scaled processes (2.8), (2.9) converge to the Moore–Penrose generalized inverse M^+ where the input matrix M is singular. Now recall that the scaled RC process (2.8), (2.9), (2.12)–(2.13) converges to the numerical generalized inverse matrix M_ϵ^+. The analysis and the estimates (including the ones for the HRC processes) can be extended provided that the 2-norms $\sigma_+^{-2}(W) = ||W^{-1}||_2$ are replaced throughout by $\sigma_{r(\epsilon)}^{-2}(W)$, where $\sigma_{r(\epsilon)}^2(W)$ is the smallest singular value of the matrix W not exceeded by ϵ. This enables various refinements from noisy perturbations of the input. Furthermore, the computation of M_ϵ^+ does not depend on whether the matrix M is singular or not. In particular, we may apply HRC processes to compute M_ϵ^+ for a positive ϵ where M is singular. If ϵ is small enough, the HRC processes output $M^+ = M_\epsilon^+$, even though the same processes may diverge if we apply them directly to M and use iteration (2.1), (2.3) or (2.8), (2.9) as a Basic Subroutine.

For the extension of the RC and HRC methods to the computation of the numerical generalized inverse M_ϵ^+ (and in particular $M^+ = M_0^+$) for a structured matrix M, an additional problem is the compression because the displacement $L(M)$ does not completely define the matrix M_ϵ^+ even for $\epsilon = 0$. For Toeplitz and Hankel matrices and for $\epsilon = 0$, the problem can be avoided [HH93], [HH94]. The following simple results solve the problem also for other classes of structured matrices wherever $\mathrm{rank}(M_\epsilon^+ M - I) = n - r_\epsilon$ is small, $r_\epsilon = \mathrm{rank}(M_\epsilon^+)$.

Theorem 11.1. *For any positive ϵ and any triple of $n \times n$ matrices A, B, and M we have*

$$\nabla_{B,A}(M_\epsilon^+) = M_\epsilon^+ A(MM_\epsilon^+ - I) - (MM_\epsilon^+ - I)BM_\epsilon^+ - M_\epsilon^+ \nabla_{A,B}(M) M_\epsilon^+.$$

Corollary 11.1. *Under the assumptions of Theorem 11.1, we have* $\mathrm{rank}(\nabla_{B,A}(M_\epsilon^+)) \leq \mathrm{rank}(\nabla_{A,B}(M)) + 2n - 2r_\epsilon$ *where* $r_\epsilon = \mathrm{rank}(M_\epsilon^+)$.

The level of the truncation of the singular values in Approach I can be defined by Corollary 11.1.

12 Concluding Remarks

We first studied Newton's iteration for matrix inversion and then more general RC (residual correction) algorithms and showed improvements in the case of structured input matrices. The algorithms based on Newton's iteration/residual correction run very fast for structured matrices, are easy to code, and allow effective parallel implementation (cf. [BP93]). We also note that the techniques discussed here (and also used in [P92], [P93], [P93a]) can be extended to other classes of structured matrices such as Cauchy-like and Vandermonde-like (Vandermonde type) matrices [P90] [PZHD97].

Our study in the second part of the thesis focused on using homotopic RC algorithms for matrix inversion. The homotopic algorithms perform matrix multiplications about as many times as the best non-homotopic RC algorithms for the inversion of unstructured positive definite matrices and substantially fewer times for complex Hermitian indefinite matrices. The homotopic RC processes (unlike their non-homotopic counterparts) generate a close initial approximation to the inverse. The latter feature supports application of the homotopic RC algorithms to the inversion of structured matrices where recursive compression of the displacements of computed approximate inverses enables every matrix multiplication at the RC steps in nearly linear time. Non-homotopic RC algorithms have no proven rapid convergence results for processes where strong compression is maintained throughout, but the results of experimental tests with Toeplitz matrices are more optimistic for the former (non-homotopic) RC algorithms than the theoretical estimates.

Appendix: Numerical Experiments

A Newton's Iteration Algorithms

A.1 Implementations of The Algorithms

Algorithm 1: Bounding the Displacement Rank of Newton's Iterates

```
%initialization
clc                                % clear command window
clear                              % clear all variables from the workspace
format long e                      % set defaults
n = 100;                           % choose size of matrix
r = 2;                             % choose r
noise = 0.001;                     % noise factor for perturbing answer matrix
rr = r + 1;                        % starting point for zeroing singular values
NZERO = zeros(n,n);                % zero nxn matrix
r0 = zeros(1,n);                   % zero row vector
Z = diag(ones(n-1,1),-1);          % downshift matrix
I = eye(n);                        % identity matrix
J = fliplr(I);                     % reversion matrix
I2 = 2*I;                          % 2*I matrix used in Newton's Iteration
%create matrix to be inverted
c1 = 2*rand(n,1)-1;                % column of random values -1 < x < 1
r1 = 2*rand(n,1)-1;                % row of random values -1 < x < 1
r1(1,1) = c1(1,1);                 % fix first element of row vector
A = toeplitz(c1,r1');              % matrix to be inverted
%generate approximation to inverse matrix
X = I/A;                           % target inverse matrix
```

```
X1 = X+(2*randn(n)-1)*noise;   % approximation matrix
        %allocate memory
rg = r0;                       % g row vector
rh = r0;                       % h row vector
DX = NZERO;                    % X (-) displacement matrix
X0 = NZERO;                    % X matrix to be converted into Y matrix
Y = NZERO;                     % Y matrix
U = NZERO;                     % U matrix for SVD
S = NZERO;                     % S matrix for SVD
V = NZERO;                     % V matrix for SVD
        %run the algorithm
disp('iterations');
disp(' ');
for i=2:10
        X0 = X1;               % set up iteration
        DX = X0-(Z'*X0*Z);     % (-) displacement operator
        [U,S,V] = svd(DX);     % singular value decomposition
        for j = rr:n
                S(j,j) = 0;    % zero unwanted singular elements
        end
        GX = U*S;              % G matrix for X
        HX = V;                % H matrix for X
        GY = GX(1:n,1:i);      % G matrix for Y
        HY = HX(1:n,1:i);      % H matrix for Y
        Y = NZERO;             % zero the Y matrix
        for j = 1:r            % recover Y matrix from [G,H]
```

```
            gi = J*GY(1:n,j:j); % g column vector
            hi = J*HY(1:n,j:j); % h column vector
            rg(1,1) = gi(1,1);  % fix first element of g row vector
            rh(1,1) = hi(1,1);  % fix first element of h row vector
            Y = Y+(toeplitz(gi,rg)'*toeplitz(hi,rh));
        end                     % Y matrix is now recovered
        X1 = Y*(I2-A*Y);        % Newton's Iteration
        R = A*X1-I;             % residual matrix
        err = norm(R,1);        % 1-norm of residual matrix
        disp(err);              % output 1-norm
end
        % housekeeping
format                          % set default output display format
        % end Algorithm 1
```

Algorithm 2: An Alternative Method of Bounding the Displacement Rank of Newton's Iterates

```
        % set up
clc                                  % clear command window
clear all                            % clear all variables from the workspace
format long e                        % set defaults
n = 100;                             % choose size of matrix
r = 2;                               % identify rank of displacement matrix
f = 1;                               % choose f
e = -1;                              % choose e
noise = 0.0001;                      % noise factor for perturbing answer matrix
disp(sprintf('noise = %0.5e\n',noise));
NZERO = zeros(n,n);                  % zero nxn matrix
Z = diag(ones(n-1,1),-1);            % downshift matrix
I = eye(n);                          % identity matrix
I2 = 2*I;                            % 2*I matrix used in Newton's Iteration
Z1F = Z;                             % 1/f-circulant matrix
Z1F(1,n) = 1/f;                      % insert factor into 1/f-circulant matrix
clear Z;                             % deallocate memory for Z
        % create matrix to be inverted
ca = 2*rand(n,1)-1;                  % column of random values -1 < x < 1
ra = 2*rand(1,n)-1;                  % row of random values -1 < x < 1
ra(1,1) = ca(1,1);                   % fix first element of row vector
A = toeplitz(ca,ra);                 % matrix to be inverted
clear ca ra;                         % deallocate memory for vectors
DA = NZERO;                          % allocate memory for displacement matrix
```

```
DA = Z1F.'*A-A*Z1F.';          % find f-circulant displacement matrix
G = DA(:,1:2);                 % generator matrix G for displacement matrix
G(n,2) = -1;                   % generator matrix G is complete
H(:,2) = -DA(n,:).';           % generator matrix H for displacement matrix
H(1,1) = 1;                    % generator matrix H is complete
clear DA;                      % deallocate memory for displacement matrix
      % generate approximation to inverse matrix
X = -I/A;                      % target inverse matrix
X0 = X+randn(n)*noise;         % approximation matrix
disp(' norm(R,1)');
disp('============');
disp(sprintf('\n initial approximation\n'));
R = A*X0+I;                    % residual matrix
err1 = norm(R,1);              % 1-norm
disp(sprintf('%0.5e\n',err1));
      % bound the displacement rank of the approximation
DX = NZERO;                    % allocate memory for displacement matrix
DX = Z1F.'*X0-X0*Z1F.';        % find f-circulant displacement matrix
clear Z1F;                     % deallocate memory for Z1F
U = NZERO;                     % allocate memory for SVD U matrix
S = NZERO;                     % allocate memory for SVD S matrix
V = NZERO;                     % allocate memory for SVD V matrix
[U,S,V] = svd(DX);             % singular value decomposition
clear DX;                      % deallocate memory for displacement matrix
rr = r+1;
for i = rr:n
```

```
        S(i,i) = 0;                 % zero unwanted singular elements
end
GX = U*S;                           % G matrix unbounded
HX = V;                             % H matrix unbounded
clear U S V;                        % deallocate memory for svd matrices
G1 = GX(1:n,1:r);                   % G matrix for DX1
H1 = HX(1:n,1:r);                   % H matrix for DX1
        % recover X1 matrix from [G,H] and XC
xc(2:n,1) = X0(1:(n-1),n);          % move X0 column n rotated into xc
xc(1) = X0(n,n);                    % fix first position of xc
xr(1,:) = flipud(X0(:,n)).';        % move X0 column n reversed,trans. into xr
ZX = toeplitz(xc,xr);               % Z_{f,lc}(X0{n-1})
X1 = NZERO;                         % zero X1 for summation
for i = 1:r
        gc(2:n,1) = G1(1:(n-1),i);          % G1 col i rotated
        gc(1) = G1(n,i);                    % fix first position
        hc = H1(:,i);                       % H1 col i
        gr(1,2:n) = flipud(G1(1:(n-1),i)).'*f;  % G1 col i reversed,scaled
        gr(1,1) = gc(1,1);                  % fix first position of row
        hr(1,2:n) = flipud(H1(2:n,i)).'/e;  % H1 col i revd.,rotated,scld.
        hr(1,1) = hc(1,1);                  % fix first position of row
        GR = toeplitz(gc,gr);               % Z_f{G1i}
        HR = toeplitz(hr,hc);               % Z_{1/e}(H1i)
        X1 = X1 + GR*HR;                    % sum over i
end
ef = e/(e-f);                       % multiplier
```

```
X1 = ZX + ef*X1;                % matrix X1 is now recovered
disp(sprintf('bounded initial approximation  n'));
R = A*X1+I;                     % residual matrix
err1 = norm(R,1);               % 1-norm
disp(sprintf('%0.5e\n',err1));
        % run the algorithm
disp(sprintf('Newton"s iterations\n'));
terr = 1e+10;                   % blow up criterion
serr = 1e-10;                   % success criterion
lerr = terr;                    % initial value of termination condition
while err1 < lerr
        AX1 = A*X1;             % approximately -I
        XX = X1*(I2+AX1);       % Newton's iteration
                % remaining references to X1 are for succeeding iteration
        G1 = XX*G;              % next G1 (sign changed)
        H1 = (H.'*XX).';        % next H1
        xc(2:n,1) = XX(1:(n-1),n); % move X1 column n rotated
        xc(1) = XX(n,n);        % fix first position of xc
        xr(1,:) = flipud(XX(:,n)).';% move X1 column n reversed,trans.
        ZX = toeplitz(xc,xr);   % Z_{f,lc}(X1{n-1})
        X1 = NZERO;             % zero X1 for summation
        for i = 1:r
                gc(2:n,1) = G1(1:(n-1),i);          % G1 col i rotated
                gc(1) = G1(n,i);                    % fix first position
                hc = H1(:,i);                       % H1 col i
                gr(1,2:n) = flipud(G1(1:(n-1),i)).'*f;  % G1 col i revd,scaled
```

```
            gr(1,1) = gc(1,1);                  % fix first row position
            hr(1,2:n) = flipud(H1(2:n,i)).'/e;  % H1 col i revd,rotd,scld
            hr(1,1) = hc(1,1);                  % fix first row position
            GR = toeplitz(gc,gr);               % Z_f(G1î}
            HR = toeplitz(hr,hc);               % Z_{1/e}(H1î)
            X1 = X1+GR*HR;                      % sum over i
        end
        X1 = ZX+ef*X1;          % matrix X1 is now recovered
        R = A*X1+I;             % residual matrix
        err0 = err1;            % save old err1
        err1 = norm(R,1);       % 1-norm
        disp(sprintf('%0.5e\n',err1));
        if err1 < serr          % if algorithm has succeeded
            lerr = err0;        % use previous error for termination
        end
end
if err1 > terr                  % if algorithm has blown up
        disp(sprintf('blow up\n')); % say so
end
        % housekeeping
format
disp(sprintf('normal program termination\n'));
        % end Algorithm 2
```

Algorithm 3: Gohberg-Semencul Residual Correction for a Toeplitz Linear System

```
%initialization
clc                              % clear command window
clear                            % clear all variables from the workspace
format long e                    % set defaults
n = 100;
s = 0;
noise = 0.001;
err = 1000;
disp('Newton"s Iteration');
disp(n);
c0 = zeros(n,1);                 % zero column vector
r0 = zeros(1,n);                 % zero row vector
r = r0;
e0 = c0;
e0(1,1) = 1;
Z = diag(ones(n-1,1),-1);        % downshift matrix
I = eye(n);                      % identity matrix
J = fliplr(I);                   % reversion matrix
%create matrix to be inverted
tc = 2*rand(n,1)-1;              % column of random values -1 < x < 1
tr = 2*rand(1,n)-1;              % row of random values -1 < x < 1
tr(1,1) = tc(1,1);               % fix first position of row
T = toeplitz(tc,tr);             % matrix to be inverted
t = zeros(n,1);                  % t vector
```

```
for i = 2:n;
        j = n+2-i;
        t(i,1) = tr(1,j);
end
t(1,1) = s;                          % first position of t vector
        %generate approximation to inverse matrix
X = I/T;                             % target inverse matrix
X0 = X+(2*randn(n)-1)*noise;         % approximation matrix
        %run the algorithm
x = X0*t;                            % initial x vector
y = X0*e0;                           % initial y vector
errX = err;
err0 = err*2;
disp('iterations = 1-norm of residual matrix');
disp(' ');
while err0 > errX
        x0 = x;                      % save old x vector
        y0 = y;                      % save old y vector
        x = x0 - X0*(T*x0-t);        % find new x vector
        y = y0 - X0*(T*y0-e0);       % find new y vector
        c = x;                       % column
        r(1,1) = c(1,1);             % fix first position of row
        X1 = toeplitz(c,r);          % L(x)
        c = Z*J*y;                   % column
        r(1,1) = c(1,1);             % fix first position of row
        X2 = toeplitz(r,c);          % LT(ZJy)
```

```
        c = y;                          % column
        r(1,1) = c(1,1);                % fix first position of row
        X3 = toeplitz(c,r);             % L(y)
        c = (Z*J*x)-e0;                 % column
        r(1,1) = c(1,1);                % fix first position of row
        X4 = toeplitz(r,c);             % LT(ZJx-e)
        X0 = X1*X2 - X3*X4;             % find new X matrix
        XR = T*X0-I;                    % find residual matrix
        err0 = errX;                    % save old error value
        errX = norm(XR,1);              % find 1-norm of residual matrix
        disp(errX);                     % output 1-norm
        if errX > err
                errX = 0;
                disp('killed by explosion detector');
        end
end
        % end Algorithm 3
```

Algorithm 4: Factor-Circulant Gohberg-Semencul Residual Correction for a Toeplitz Linear System

```
%initialization
clc                          % clear command window
clear                        % clear all variables from the workspace
format long e                % set defaults
n = 100;                     % choose size of matrix
e = -1;
f = 1;
s = 0;
link = 0;
noise = 0.001;               % noise factor for perturbing answer matrix
err = 10000;
c0 = zeros(n,1);             % zero column vector
r0 = zeros(1,n);             % zero row vector
e0 = c0;
e0(1,1) = 1;
Z = diag(ones(n-1,1),-1);    % downshift matrix
I = eye(n);                  % identity matrix
J = fliplr(I);               % reversion matrix
%create matrix to be inverted
c = 2*rand(n,1)-1;           % column of random values -1 < x < 1
r = 2*rand(1,n)-1;           % row of random values -1 < x < 1
r(1,1) = c(1,1);             % fix first position of row
T = toeplitz(c,r);           % matrix to be inverted
t = zeros(n,1);              % t(e,s) vector
```

```
for i = 2:n;
        j = n+2-i;
        t(i,1) = c(i,1)-e*r(1,j);
end
t(1,1) = s;                          % first position of t(e,s) vector
        %generate approximation to inverse matrix
X = I/T;                             % target inverse matrix
X0 = X+(2*randn(n)-1)*noise;         % approximation matrix
        %run the algorithm
r = r0;
c = c0;
x = X0*t;                            % initial x vector
y = X0*e0;                           % initial y vector
ef = 1-e*f;                          % loop invarient
efe = ef*e0;                         % loop invarient vector
errX = err;
err0 = err*2;
err1 = err0;
disp('iterations = 1-norm of residual matrix');
disp(' ');
disp('-new norm- -prev norm- -prev sqr- —ratio—-');
done = 0;
while done == 0
        x0 = x;                      % save old x vector
        y0 = y;                      % save old y vector
        x = x0 - X0*(T*x0-t);        % find new x vector
```

```
y = y0 - X0*(T*y0-e0);          % find new y vector
c = y;                          % column
for i = 2:n;
      j = n+2-i;
      r(1,i) = f*c(j,1);
end
r(1,1) = c(1,1);                % fix first position of row
X1 = toeplitz(c,r);             % Zf(y)
c = x;                          % column
for i = 2:n;
      j = n+2-i;
      r(1,i) = c(j,1)/e;
end
r(1,1) = c(1,1);                % fix first position of row
X2 = toeplitz(c,r);             % Z1/e(x)
c = x-efe;                      % column
for i = 2:n;
      j = n+2-i;
      r(1,i) = f*c(j,1);
end
r(1,1) = c(1,1);                % fix first position of row
X3 = toeplitz(c,r);             % Zf(x-(1-ef)e0)
c = y;                          % column
for i = 2:n;
      j = n+2-i;
      r(1,i) = c(j,1)/e;
```

```
        end
        r(1,1) = c(1,1);                % fix first position of row
        X4 = toeplitz(c,r);             % Z1/e(y)
        X0 = (X1*X2-X3*X4)/ef;          % find new X matrix
        XR = T*X0-I;                    % find residual matrix
        err0 = errX;                    % save old error value
        errX = norm(XR,1);              % find 1-norm of residual matrix
        if link == 1
                sqr = err0*err0;
                dif = errX/sqr;
                disp(sprintf('%0.5e %0.5e %0.5e %0.5e',errX,err0,sqr,dif));
        else
                link = 1;
                dif = 0;
                disp(sprintf('%0.5e',errX));
        end
        if dif > 1
                done = 1;
        end
        if errX > err
                done = 1;
                disp('killed by explosion detector');
        end
end
```

A.2 Newton's Iteration Algorithms Tests

Algorithms 1-4 were tested using MATLAB, a software package marketed by The Math Works, Inc., of Massachusetts. In the first group of experiments, the input matrices A were generated as 100×100 Toeplitz matrices whose first columns and first rows were filled with random entries chosen from a uniform distribution on the interval $-1 < \alpha_{i,j} < 1$. To generate an initial approximation X_0 to $-A^{-1}$, we first numerically computed the actual inverse matrix $-X$ and then perturbed its entries by adding some small random values. To generate the perturbations, we first composed 100×100 matrices with random entries chosen under a normal distribution with mean zero and variance one, and then scaled the values by a noise factor. We used a noise factor of 0.001 for Algorithms 1, 3, and 4. Algorithm 2 requires a closer initial approximation to the solution to yield quadratic convergence, so we decreased the noise factor in its test to 0.0001. Once the noise matrix was obtained, we added it to the actual inverse.

We then tested the algorithms, running a group of forty tests for each algorithm. In most cases rapid convergence was achieved, with four exceptions in the second group, one exception in the third group, and one exception in the fourth group. To test the convergence rate, we computed the column norm $p_i = ||AX_i + I||_1$ of the residual matrix $R_i = AX_i + I$ obtained in the ith iteration step. Sample test results are presented in Tables A.1–A.7. We also supplied the value

$$\text{cond}_1(A) = ||A||_1||A^{-1}||_1.$$

We let $f = -e = 1$ in Algorithms 2 and 4, and we performed Algorithms 3 and 4 for vectors $\mathbf{t}$ with $s = 0$.

The results of these computations fell into three classes. The first class, represented in Tables A.1–A.4, shows the computations where rapid convergence was observed immediately. The second class, represented in Tables A.5–A.7, shows the

computations with the input matrices for which iteration initially stumbled and then, after a certain number of steps, s, started to converge rapidly. In fact, we observed only four cases out of forty in the tests for Algorithm 2 ($s \leq 4$), three cases out of forty in the tests of Algorithm 3 ($s = 2$) and one case out of forty in the tests of Algorithm 4 ($s = 3$). In the third class of inputs, the iteration showed no sign of convergence even after 10 Newton steps. Only six such cases occured in all 160 tests. We display the results for 3 samples in the first class and 2 samples in the second class, in which case we displayed the p_i after s steps of stumbling. In all tables, the results are displayed only until convergence, up to the step after which the roundoff errors started to exceed the approximation errors.

In some experiments the iteration seemed to converge irregularly, in the sense that the residual norm exceeded 1 initially but the iteration still converged. The apparent reason is that the bound

$$||I + AX_{i+1}|| \leq ||I + AX_i||^2$$

can be strict inequality, rather than equation.

On the other hand, the residual norm sometimes decreased slower than quadratically. This is immediately explained by the influence of the stage of adjusting the matrices to Toeplitz format, for Algorithms 3 and 4, to the selected Toeplitz-like format, for Algorithm 2, and to their representation with shorter displacement generators, for Algorithm 1. With these comments in mind, the test results are quite consistent with the theoretical estimates for the convergence rate.

Table A.1: Tests for inputs of class 1 for Algorithm 1.

	Sample Test 1	Sample Test 2	Sample Test 3
$\text{cond}_1(A)$	2.024643e+004	1.365480e+005	2.223065e+004
r_0	1.932960e–002	1.990289e–002	2.206516e–002
r_1	5.822783e–005	3.102183e–005	8.922756e–005
r_2	9.966761e–011	1.277523e–010	8.006189e–010
r_3	8.220646e–013	8.420042e–012	1.920519e–012

Table A.2: Tests for inputs of class 1 for Algorithm 2.

	Sample Test 1	Sample Test 2	Sample Test 3
$\text{cond}_1(A)$	3.274079e+003	5.836559e+003	3.425782e+003
r_0	4.392883e–003	2.053855e–002	1.876769e–002
r_1	4.999007e–005	3.952711e–003	2.661747e–003
r_2	9.924380e–009	3.011774e–004	7.458110e–005
r_3	2.121324e–012	9.710310e–007	1.452111e–007
r_4		2.223482e–011	2.891295e–011

Table A.3: Tests for inputs of class 1 for Algorithm 3.

	Sample Test 1	Sample Test 2	Sample Test 3
$\text{cond}_1(A)$	3.032816e+003	7.932677e+003	2.836675e+003
r_0	7.205510e-001	7.046618e-001	1.330095e-001
r_1	6.074104e-002	7.454610e-002	3.528652e-003
r_2	8.190064e-004	1.608024e-003	3.643328e-006
r_3	2.160530e-007	1.015745e-006	4.971280e-012
r_4	7.357363e-013	4.453048e-012	

Table A.4: Tests for inputs of class 1 for Algorithm 4.

	Sample Test 1	Sample Test 2	Sample Test 3
$\text{cond}_1(A)$	1.727930e+003	2.435048e+003	3.248792e+003
r_0	2.246859e-001	5.085527e-001	6.194036e-001
r_1	9.875483e-003	4.438413e-002	7.533096e-002
r_2	2.864990e-005	4.116238e-004	1.865356e-003
r_3	2.354060e-010	4.307276e-008	1.342796e-006
r_4	1.764283e-013	3.781741e-013	1.848549e-012

Table A.5: Tests for inputs of class 2 for Algorithm 2.

	Sample Test 1	Sample Test 2	Sample Test 3
$\text{cond}_1(A)$	4.637187e+003	5.763326e+003	1.349509e+003
s	4	2	1
r_0	6.170632e-003	1.021014e-003	9.937866e-003
r_1	7.897623e-004	5.588185e-005	1.241422e-005
r_2	1.426298e-005	1.621575e-007	2.137849e-009
r_3	4.577198e-009	7.528647e-012	6.846061e-012
r_4	2.713013e-011		

Table A.6: Tests for inputs of class 2 for Algorithm 3.

	Sample Test 1	Sample Test 2	Sample Test 3
$\text{cond}_1(A)$	2.042839e+004	1.483413e+004	1.044744e+004
s	3	2	2
r_0	3.699548e-001	9.831201e-002	8.105060e-002
r_1	6.345701e-002	4.084126e-003	2.120317e-003
r_2	1.766525e-003	7.245445e-006	1.247355e-006
r_3	1.372439e-006	2.563352e-011	3.775972e-011
r_4	3.605207e-011		

Table A.7: Tests for inputs of class 2 for Algorithm 4.

	Sample Test 1
$\text{cond}_1(A)$	2.000268e+004
s	3
r_0	1.097395e-001
r_1	5.196420e-003
r_2	1.226878e-005
r_3	8.729462e-011

B Homotopic Algorithms

The presented Newton-Structured Iteration algorithms (that is, structured RC processes for $p = 2$) were tested numerically for $n \times n$ Toeplitz input matrices M. The compression was achieved by means of the truncation of the singular values (according to Approach I) and was implemented as Algorithm 1 above. The tests were performed by M. Kunin at the Graduate Center of CUNY, in cooperation with the author.

The tests used the following computational facilities:

- OS – Red Hat Linux 7.0
- compiler – GCC 2.96 (also using bundled random number generator)
- library – CLAPACK 3.0 (routines for computing SVD of real and complex matrices and eigenvalues of symmetric matrices)

Both non-homotopic and stiff homotopic versions of Newton-Structured Iteration were applied to the same input matrices M. In non-homotopic processes the compression level, that is, the number l of untruncated singular values of the displacements, was chosen adaptively to minimize l as long as convergence was achieved. More precisely, for a fixed threshold value ϵ, the candidate compression level l was calculated as follows. Let $\sigma_1^{(i)}, \ldots, \sigma_n^{(i)}$ denote the singular values of X_i written in the non-increasing order. Then we chose $l = l(i)$ satisfying $\sigma_{l+1}^{(i)} < \epsilon\sigma_1^{(i)}$, $\sigma_l^{(i)} \geq \epsilon\sigma_1^{(i)}$. If Newton's process diverged for these ϵ and l, then all results of the computation (obtained by this moment) were discarded, except that the contribution to the overall work of the iteration was counted. Then the iteration process was repeated with the compression level $l/2$. In the case of divergence, this value was recursively halved further until convergence but at most 10 times. The experiments for homotopic processes were limited to recursive optimization

of the tolerance values θ_h and consequently the homotopic step sizes, under a stiff 0.7-down policy. According to this policy, the initial value of θ was always set to 0.7 and never increased. In the case of divergence, θ was recursively halved, and the process was repeated until convergence. For a fixed θ, the step sizes were calculated using LAPACK for computing the eigenvalues of the matrix M. The value l of the compression level was fixed and remained invariant in all Newton's steps throughout the entire homotopic process. This value was chosen experimentally when convergence with this value was observed for one or two test runs. This was our simplified preliminary policy, subject to improvement in our future experiments.

The computations stopped at the final homotopic and non-homotopic steps where the residual norm decreased to the single precision 0; at all other homotopic steps, the computations stopped where the residual norm decreased below 10^{-6}. (The latter bound was a little smaller than was necessary for convergence.)

The algorithm was tested for $n \times n$ Toeplitz matrices M of the following classes (see details below).

1. Real symmetric tridiagonal Toeplitz matrices $(t_{i,j})_{i,j=0}^{n-1}$, $t_{i,j} = 0$ where $|i-j| > 1$, $t_{i+1,i} = t_{i,i+1} = 1$, $t_{i,i}$ equals 4 or -2.
2. The matrices $\left(\frac{1}{1+|i-j|}\right)_{i,j=0}^{n-1}$.
3. Randomly generated Toeplitz matrices.
4. Randomly generated real symmetric positive definite matrices with a specified condition number.
5. Randomly generated real symmetric indefinite Toeplitz matrices.

The tests results were differentiated further according to the condition number of the matrix M and the size $n \times n$, for n ranging from 50 to 350.

An $n \times n$ random real symmetric Toeplitz matrix of class 5 was defined by generating the n random entries of its first row; for an unsymmetric Toeplitz matrix of class 3, also the $n-1$ remaining entries of its first column were generated. The random entries were generated as the random normally distributed variables with the mean 0 and the standard deviation 1. At this stage, a random number generator was applied with the **rand()** function from the standard C library that comes with the GCC compiler for Cygwin on Windows 2000. The condition number was most frequently quite small for random matrices, and then the algorithms converged very rapidly. To make the results more meaningful, part of the experiments was restricted to the matrices with larger condition numbers. To form a matrix of class 4, that is, to achieve positive definiteness and a desired condition number, we computed the two extremal eigenvalues of a random real symmetric Toeplitz matrix and then added the matrix aI for an appropriate positive a.

For unsymmetric and symmetric indefinite Toeplitz matrices M, the non-homotopic Newton's process was initialized with the matrices

$$X_0 = M^T/(\|M\|_1\|M\|_\infty).$$

For a symmetric positive definite matrix M, the same process was applied with $X_0 = I/\|M\|_F$. For the homotopic processes with the same symmetric input matrices M, the same Newton's processes were applied with the invariant truncation level $l = 2$, except that the initial matrices X_0 were determined by the homotopic rules and the choice of the matrix M_0. For unsymmetric input matrices M, both homotopic and non-homotopic processes also were applied to two symmetrized matrices M^TM and $\left(\begin{smallmatrix} 0 & M \\ M^T & 0 \end{smallmatrix}\right)$ (cf. (8.1) and (8.2)). For homotopic processes, in these two cases, the invariant truncation levels $l = 12$ and $l = 6$ were selected, respectively. In the symmetric indefinite case, the initial choice (8.3) was used for the matrix M_0. In the positive definite case, the matrix M_0 was selected according

to (7.2). The initial threshold bound ϵ in non-homotopic Newton's processes was selected at the levels 0.025 for the unsymmetric and symmetric indefinite matrices, 0.00005 for the symmetrized matrices $\left(\begin{smallmatrix} 0 & M \\ M^T & 0 \end{smallmatrix}\right)$.

In the column K we show the number of test runs where the random input matrices M have the condition numbers bounded from below by the value in the respective line of the column κ. The columns N and L show the number of Newton's steps and the overall work (that is, the sum of the truncation levels in all Newton's steps) in non-homotopic processes; the columns N_H and L_H show the same data for homotopic processes. H is the number of homotopic steps, s_θ is the number of times the tolerance value θ was halved (split) during the homotopic process. All these data are averaged over the K test runs for the selected range of the condition number, except that for symmetric positive definite matrices M the column κ shows the fixed condition numbers $\kappa(M)$. For non-random matrices M of classes 1 and 2, we have $K = 1$, that is, the tests run for a single fixed matrix M. For each class of input matrices, the truncation level for the singular values was decreased recursively as long as this did not destroy the convergence in experimental tests.

In addition to the data shown in Tables B.1–B.16, we include a table showing an interesting phenomenon observed in more than 25% test runs for non-homotopic processes with the random Toeplitz input matrices of class 3. That is, in these test runs the residual 2-norm was growing above 1 (and sometimes well beyond 1) and then decreased down to 0 in the next Newton's steps. This phenomenon must be attributed to compression because otherwise the residual 2-norm would have been squared in each Newton's step, which would have automatically implied the divergence of Newton's iteration. Table B.17 demonstrates this phenomenon in some details. That is, for each test run it shows the condition number of the input matrix M and the maximum residual norm ρ_i reached during the iteration

process, which finally converged. The columns marked by ρ_1, ρ_2, and ρ_3 show the values of the residual norm for non-homotopic process with no symmetrization of M and with symmetrization based on (8.2) and (8.1), respectively.

Table B.1: Results for the tridiagonal Toeplitz matrices M of class 1 with the entries on the main diagonal equal to 4 and on the first super and subdiagonals equal to 1.

n	κ	H	N_H	L_H	N	L
50	2.992	3	21	42	8	16
100	2.998	3	20	40	9	18
150	2.999	3	21	42	9	18
200	3.000	3	20	40	9	18
250	3.000	3	20	40	9	18
300	3.000	3	21	42	9	18
350	3.000	3	20	40	10	20

Table B.2: Results for the tridiagonal Toeplitz matrices M of class 1 with the entries on the main diagonal equal to 2 and on the first super and subdiagonals equal to -1.

n	κ	H	N_H	L_H	N	L
50	1053.480	8	46	92	16	32
100	4133.640	9	56	112	19	38
150	9240.230	9	56	112	20	40
200	16373.200	10	59	118	21	42
250	25532.700	10	60	120	22	44
300	36718.500	11	63	126	23	46
350	49930.800	11	67	134	23	46

Table B.3: The matrices $M = \left(\frac{1}{|i-j|}\right)_{i,j}$ of class 2

n	κ	H	N_H	L_H	N	L
50	16.221	4	25	50	9	18
100	19.642	4	25	50	9	18
150	21.680	4	26	52	10	20
200	23.138	4	27	54	10	20
250	24.274	4	26	52	10	20
300	25.205	4	27	54	10	20

Table B.4: Random Toeplitz matrices M of class 3, $n = 100$ (non-homotopic iteration)

κ	K	ϵ	N	L
10000	1	0.025	32.00	124.00
5000	2	0.013	61.00	261.00
1000	7	0.022	41.71	185.86
500	14	0.018	42.36	187.36
100	69	0.025	23.83	101.96
50	56	0.025	20.91	88.86
10	45	0.025	18.82	76.11

κ	H	N_H	L_H	s_θ	ϵ	N	L
10000	10.00	67.00	402.00	0.00	6.3e-04	102.00	1115.00
5000	19.50	112.50	675.00	1.00	1.3e-03	63.00	695.00
1000	22.00	119.86	719.14	1.43	2.1e-03	44.00	497.57
500	16.64	97.36	584.14	1.36	2.4e-03	29.64	329.14
100	13.57	82.46	494.78	1.52	2.5e-03	23.64	276.00
50	11.39	70.34	422.04	1.61	2.5e-03	20.93	249.79
10	9.40	56.96	341.73	1.49	2.5e-03	18.76	220.82

Table B.5: Symmetric positive definite Toeplitz matrices M^TM for a random Toeplitz matrix M of class 3, $n = 100$ (the column K is the same as in Table B.4) (Note: in this case no decrease of the initial tolerance value $\theta = 0.7$ was required in the experiments, so the resulting table does not contain the s_θ column)

κ	H	N_H	L_H	ϵ	N	L
10000	18.00	106.00	1272.00	1.3e-05	84.00	690.00
5000	16.50	99.50	1194.00	2.8e-05	65.00	584.00
1000	14.71	88.00	1056.00	1.4e-05	79.71	737.29
500	12.50	76.14	913.71	3.6e-05	42.64	379.50
100	10.42	63.57	762.78	4.8e-05	23.62	214.30
50	8.86	54.70	656.36	5.0e-05	19.12	180.95
10	7.69	47.13	565.60	5.0e-05	16.73	162.96

Table B.6: Results for random symmetric positive definite Toeplitz matrices M of class 4, n=50

κ	K	H	N_H	L_H	ϵ	N	L
250000	19	12	73.16	146.32	0.050	24.00	49.58
10000	19	9	56.42	112.84	0.050	19.74	40.95
5000	19	9	54.63	109.26	0.048	19.89	42.74
1000	20	8	47.40	94.80	0.050	16.00	33.85
500	20	7	43.25	86.50	0.050	15.10	31.40
100	18	6	36.28	72.56	0.050	12.94	26.56
50	20	5	31.70	63.40	0.050	12.00	24.15
10	19	4	25.89	51.79	0.050	9.63	19.26

Table B.7: Results for random symmetric positive definite Toeplitz matrices M of class 4, n=100

κ	K	H	N_H	L_H	ϵ	N	L
250000	19	12	73.68	147.37	0.048	26.11	54.53
10000	17	9	56.24	112.47	0.044	24.06	53.24
5000	19	9	54.21	108.42	0.038	28.00	66.16
1000	14	8	47.64	95.29	0.048	18.36	38.93
500	19	7	43.53	87.05	0.049	16.74	35.47
100	16	6	36.50	73.00	0.050	13.19	27.88
50	19	5	31.84	63.68	0.050	12.11	24.68
10	18	4	25.72	51.44	0.050	10.00	20.00

Table B.8: Results for random symmetric positive definite Toeplitz matrices M of class 4, n=150

κ	K	H	N_H	L_H	ϵ	N	L
250000	15	12	73.53	147.07	0.036	40.00	97.53
10000	11	9	56.09	112.18	0.044	24.18	53.82
5000	14	9	54.64	109.29	0.035	30.36	72.29
1000	17	8	47.65	95.29	0.049	18.06	38.53
500	18	7	43.39	86.78	0.049	17.56	37.72
100	13	6	36.54	73.08	0.050	13.85	29.46
50	18	5	32.11	64.22	0.050	12.94	26.39
10	18	4	25.39	50.78	0.050	10.00	20.00

Table B.9: Results for random symmetric positive definite Toeplitz matrices M of class 4, n=200

κ	K	H	N_H	L_H	ϵ	N	L
250000	12	12	73.50	147.00	0.035	39.17	93.33
10000	11	9	56.82	113.64	0.048	21.64	45.45
5000	12	9	54.17	108.33	0.038	29.75	70.33
1000	13	8	47.46	94.92	0.035	30.23	72.23
500	13	7	43.54	87.08	0.044	20.38	45.77
100	15	6	36.53	73.07	0.050	14.40	29.67
50	12	5	32.00	64.00	0.050	13.17	26.75
10	12	4	25.58	51.17	0.050	10.17	20.33

Table B.10: Results for random symmetric indefinite Toeplitz matrices M of class 5, n=50

κ	K	H	N_H	L_H	s_θ	ϵ	N	L
1000	6	8.00	50.00	100.00	0.00	0.015	52.00	235.17
500	9	7.00	45.00	90.00	0.00	0.018	39.89	179.00
100	46	6.33	40.41	80.83	0.00	0.022	28.67	126.48
50	25	5.28	35.36	70.72	0.04	0.025	20.00	83.76
10	14	4.93	32.21	64.43	0.00	0.025	17.50	69.64

Table B.11: Results for random symmetric indefinite Toeplitz matrices M of class 5, n=100

κ	K	H	N_H	L_H	s_θ	ϵ	N	L
10000	2	10.00	60.50	121.00	0.00	0.019	45.50	204.00
5000	3	9.00	57.00	114.00	0.00	0.003	118.00	605.67
1000	7	8.14	50.71	101.43	0.00	0.013	64.71	336.29
500	9	7.00	46.11	92.22	0.00	0.018	42.89	209.22
100	64	6.67	43.05	86.09	0.09	0.022	29.22	136.97
50	12	6.58	43.83	87.67	0.42	0.025	21.00	93.50
10	2	5.00	33.50	67.00	0.00	0.025	19.50	78.50

Table B.12: Results for random symmetric indefinite Toeplitz matrices M of class 5, n=150

κ	K	H	N_H	L_H	s_θ	ϵ	N	L
10000	8	10.25	64.12	128.25	0.00	0.015	83.00	437.00
5000	3	9.00	59.33	118.67	0.00	0.010	88.00	479.00
1000	22	8.50	54.68	109.36	0.09	0.012	75.41	406.23
500	15	7.20	49.40	98.80	0.07	0.019	42.00	212.47
100	45	7.11	45.67	91.33	0.24	0.022	30.22	147.13
50	5	5.80	38.80	77.60	0.20	0.022	26.00	118.40
10	1	5.00	35.00	70.00	0.00	0.025	20.00	84.00

Table B.13: Results for random symmetric indefinite Toeplitz matrices M of class 5, n=200

κ	K	H	N_H	L_H	s_θ	ϵ	N	L
10000	2	11.00	71.00	142.00	0.00	0.001	194.00	1150.50
5000	2	9.00	60.00	120.00	0.00	0.016	63.50	330.50
1000	19	8.47	54.58	109.16	0.11	0.012	67.32	356.74
500	25	8.68	56.68	113.36	0.48	0.018	44.48	228.84
100	48	7.50	48.35	96.71	0.33	0.023	30.44	149.83
50	1	6.00	38.00	76.00	0.00	0.025	22.00	100.00
10	1	5.00	33.00	66.00	0.00	0.025	20.00	89.00

Table B.14: Results for random symmetric indefinite Toeplitz matrices M of class 5, n=250

κ	K	H	N_H	L_H	s_θ	ϵ	N	L
10000	6	13.00	77.83	155.67	0.33	0.005	111.33	634.17
5000	4	9.00	61.00	122.00	0.00	0.003	150.00	921.50
1000	30	9.70	61.33	122.67	0.50	0.016	56.97	293.50
500	17	8.59	57.29	114.59	0.53	0.015	52.29	281.18
100	40	8.28	52.67	105.35	0.57	0.020	35.98	185.47

Table B.15: Results for random symmetric indefinite Toeplitz matrices M of class 5, n=300

κ	K	H	N_H	L_H	s_θ	ϵ	N	L
10000	5	13.20	83.00	166.00	0.60	0.004	139.60	871.80
5000	4	12.50	75.75	151.50	0.50	0.011	82.50	457.00
1000	26	9.96	62.85	125.69	0.50	0.013	68.85	382.31
500	32	9.12	59.44	118.88	0.62	0.017	49.25	258.75
100	28	8.54	53.86	107.71	0.64	0.021	35.64	182.25
50	1	6.00	40.00	80.00	0.00	0.025	22.00	97.00

Table B.16: Results for homotopic and nonhomotopic processes applied to symmetric positive definite matrix obtained from the original by premultiplying with its transpose. (Note: in this case no decrease of the initial tolerance value $\theta = 0.7$ was required in the experiments, so the resulting table does not contain the s_θ column)

κ	H	N_H	L_H	ϵ	N	L
10000	18.00	106.00	1272.00	1.3e-05	84.00	690.00
5000	16.50	99.50	1194.00	2.8e-05	65.00	584.00
1000	14.71	88.00	1056.00	1.4e-05	79.71	737.29
500	12.50	76.14	913.71	3.6e-05	42.64	379.50
100	10.42	63.57	762.78	4.8e-05	23.62	214.30
50	8.86	54.70	656.36	5.0e-05	19.12	180.95
10	7.69	47.13	565.60	5.0e-05	16.73	162.96

Table B.17: Maximal values of the residual norms for non-homotopic processes with the original, symmetrized double-sized, and symmetrized positive definite matrices, respectively. Whenever the value did not exceed 1, it is denoted by ****** in the table.

κ	ρ_1	ρ_2	ρ_3
13117.400	1.01437	1.24521	2.29178
9997.910	1.00164	5.04473	2.87706
6038.480	1.00109	******	1.00278
3859.550	2.65327	3.61647	6.60274
2610.770	1.04845	2.71020	1.79991
2464.170	1.01631	1.00020	******
2324.950	2.43046	2.42135	1.49457
1848.360	1.00955	1.00033	1.13965
1597.240	1.00009	******	4.52156
1188.030	1.19201	******	1.57259
907.966	1.00097	1.01116	******
856.188	1.00250	******	1.10969
841.877	1.00008	******	1.88333
788.226	1.00240	******	******
740.068	1.00006	4.63459	1.49978
695.662	1.00002	******	******
689.868	1.00671	2.39270	3.50735
627.683	1.00044	******	2.46652
616.774	1.00332	******	1.18659
575.609	******	******	5.43561
554.912	1.17903	1.46120	3.31692
514.378	1.08366	******	******
512.452	1.06951	******	******
508.758	1.00109	******	4.44255
487.164	1.00206	******	1.49436
472.963	1.01446	******	1.61860
465.666	1.00001	******	******
441.397	1.00312	******	1.63125
428.375	1.00073	******	******
341.934	1.25551	******	1.01262

κ	ρ_1	ρ_2	ρ_3
340.355	1.00045	******	******
302.378	1.05618	******	1.10584
282.635	1.00102	******	******
278.193	1.16907	******	******
264.911	1.02277	******	******
261.404	1.00168	******	******
259.630	1.00628	******	2.75387
252.902	1.00218	******	1.22844
248.518	1.00040	******	1.69287
243.424	1.00110	******	******
242.588	2.09438	******	******
228.158	1.10735	******	******
223.702	1.01272	******	1.21397
202.115	1.00013	******	******
200.249	1.00212	******	******
197.326	1.00032	******	******
196.148	2.36707	******	******
190.886	1.00005	******	******
190.090	1.00010	******	2.96128
188.992	1.00323	******	******
174.938	1.00159	******	******
172.100	1.00260	******	******
171.442	1.05699	******	******
171.416	1.00030	******	******
166.156	1.00021	******	******
165.404	1.00381	******	******
162.923	1.00054	******	******
160.472	1.00127	******	******
158.845	1.00001	******	******
158.335	1.00005	******	******
147.662	1.00834	******	******
143.380	1.00044	******	******
141.694	1.00001	******	******
136.384	1.00002	******	******

κ	ρ_1	ρ_2	ρ_3
132.914	1.00011	******	******
128.667	1.00001	******	******
125.703	1.00001	******	******
114.952	1.00022	******	******
102.699	1.00023	******	******
69.764	1.00001	******	******
61.406	1.19951	******	******
58.648	1.87989	******	******

References

[A] K. E. Atkinson, *An Introduction to Numerical Analysis,* Wiley, New York, 1978.

[AGr88] G.S. Ammar, W.B. Gragg, Supperfast Solution of Real Positive Definite Toeplitz Systems, *SIAM J. on Matrix Anal. Appl.,* **9**, **1**, 61–76, 1988.

[AG89] G. S. Ammar, P. Gader, New Decompositions of the Inverse of a Toeplitz Matrix, *Proc. 1989 Int. Symp. on Math Theory of Networks and Systems (MTNS'89)*, Amsterdam, 1989.

[B68] C.A. Boyer, *A History of Mathematics*, Wiley, New York, 1968.

[BGR90] J. A. Ball, I. Gohberg, L. Rodman, Interpolation of Rational Matrix Functions, *Operator Theory: Advances and Applications*, **45**, Birkhäuser, Basel, 1990.

[BGR90a] J. A. Ball, I. Gohberg, L. Rodman, Nehari Interpolation Problem for Rational Matrix Functions: the Generic Case, H_∞*-control Theory* (E. Mosca, L. Pandolfi, editors), 277–308, Springer, Berlin, 1990.

[B-I66] A. Ben-Israel, A Note on Iterative Method for Generalized Inversion of Matrices, *Math. Comp.*, **20**, 439–440, 1966.

[B-IC66] A. Ben-Israel, D. Cohen, On Iterative Computation of Generalized Inverses and Associated Projections, *SIAM J. Numer. Anal.*, **3**, 410–419, 1966.

[BM,a] D. A. Bini, B. Meini, Approximate Displacement Rank and Applications, preprint.

[BP93] D. A. Bini, V. Y. Pan, Improved Parallel Computations with Toeplitz-like and Hankel-like Matrices, *Linear Algebra and Its Applications*, **188/189**, 3–29, 1993.

[BP94] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[BP98] D. Bini, V. Y. Pan, Computing Matrix Eigenvalues and Polynomial Zeros Where the Output is Real, *SIAM J. on Computing*, **27**, **4**, 1099–1115, 1998.

[Bun] J.R. Bunch, Stability of Methods for Solving Toeplitz Systems of Equation, *SIAM J. Sci. Stat. Comput.*, **6**, **2**, 349–364, 1985.

[CKL-A87] J. Chun, T. Kailath, H. Lev-Ari, Fast Parallel Algorithm for QR-factorization of Structured Matrices, *SIAM Journal on Scientific and Statistical Computing*, **8**, **6**, 899–913, 1987.

[CN96] R. H. Chan, M. K. Ng, Conjugate Gradient Methods for Toeplitz Systems, *SIAM Review*, **38**, 427–482, 1996.

[CN99] R. H. Chan, M. K. Ng, Iterative Methods for Linear Systems with Matrix Structure, *Fast Reliable Algorithms for Matrices with Structure*, (T. Kailath, A.H. Sayed, Editors) 117–152, SIAM Publications, Philadelphia, 1999.

[CPVBWa] G. Codevico, V. Y. Pan, M. Van Barel, X. Wang, Iterative Inversion of Structured Matrices, Preliminary Report, 2002.

[EPY98] I. A. Emiris, V. Y. Pan, Y. Yu, Modular Arithmetic for Linear Algebra Computations in the Real Field, *J. of Symbolic Computation*, **26**, 71–87, 1998.

[FF63] D. K. Faddeev, V. N. Faddeeva, *Computational Methods of Linear Algebra,* W. H. Freeman, San Francisco, 1963.

[G] M. Gu, Stable and Efficient Algorithms for Structured Systems of Linear Equations, *SIAM J. on Matrix Anal. Appl.*, 1997.

[GE95] M. Gu, S. C. Eisenstat, A Divide-and-Conquer Algoritm for the Symmetric Tridiagonal Eigenproblem, *SIMAX*, **16**, **1**, 172–191, 1995.

[GKO95] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian Elimination With Partial Pivoting for Matrices with Displacement Structure, *Math. of Computation,* **64**, 1557–1576, 1995.

[GL96] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins Univeristy Press, Baltimore, Maryland, 1989 (2nd edition), 1996 (3rd edition).

[GO92] I. Gohberg, and V. Olshevsky, Circulant, Displacements and Decomposition of Matrices, *Integral Equations and Operator Theory*, **15**, **5**, 730–743, 1992.

[GO94] I. Gohberg, V. Olshevsky, Complexity of Multiplication with Vectors for Structured Matrices, *Linear Algebra Appl.*, **202**, 163–192, 1994.

[GO94b] I. Gohberg, V. Olshevsky, Fast State Space Algorithms for Matrix Nehari and Nehari-Takagi Interpolation Problems, *Integral Equations and Operator Theory*, **20**, **1**, 44–83, 1994.

[GS72] I. Gohberg, A. Semencul, On the Inversion of Finite Toeplitz Matrices and Their Continous Analogs, *Mat. Issled.*, **2**, 187–224, 1972.

[H95] G. Heinig, Inversion of Generalized Cauchy Matrices and the Other Classes of Structured Matrices, *Linear Algebra for Signal Processing, IMA Volume in Math. and Its Applications,* **69**, 95–114, Springer, 1995.

[HH93] G. Heinig, F. Hellinger, On the Bezoutian Structure of the Moore–Penrose Inverses of Hankel Matrices, *SIAM J. on Matrix Analysis and Applications*, **14**, **3**, 629–645, 1993.

[HLPW86] M. T. Heath, A. J. Laub, C. C. Paige, R. C. Ward, Computing the Singular Value Decomposition of a Product of Two Matrices, *SIAM J. of Scientific and Statistical Computing*, **7**, **4**, 1147–1159, 1986.

[HH94] G. Heinig, F. Hellinger, Moore–Penrose Generalized Inverse of Square Toeplitz Matrices, *SIAM J. on Matrix Analysis and Applications*, **15**, **2**, 418–450, 1994.

[HR84] G. Heinig, K. Rost, *Algebraic Methods for Toeplitz-like Matrices and Operators, in Operator Theory: Advances and Applications*, (I. Gohberg editor), **13**, Birkhäuser, Basel, 1984.

[IK66] E. Issacson, H. B. Keller, *Analysis of Numerical Methods*, Wiley, New York, 1966.

[KKM79] T. Kailath, S. Y. Kung, M. Morf, Displacement Ranks of Matrices and Linear Equations, *J. Math. Anal. Appl.*, **68**, **2**, 395–407, 1979.

[KS99] T. Kailath, A. H. Sayed (Editors), *Fast Reliable Algorithms for Matrices with Structure*, SIAM Publications, Philadelphia, 1999.

[OP98] V. Olshevsky, V. Y. Pan, A Unified Superfast Algorithm for Boundary Rational Tangential Interpolation Problem, *Proc. 39th Ann. IEEE Symp. Foundations of Comp. Sci.*, 192–201, IEEE Comp. Soc. Press, 1998.

[P90] V. Y. Pan, On Computations with Dense Structured Matrices, *Math. Comp.*, **55**, **191**, 179-190, 1990. Proceeding version: *Proc. Intern. Symp. on Symbolic and Algebraic Comp. (ISSAC'89)*, 34–42, ACM Press, New York, 1989.

[P92] V. Y. Pan, Parallel Solution of Toeplitz-like Linear Systems, *J. of Complexity*, **8**, 1–21, 1992.

[P92a] V. Y. Pan, Parametrization of Newton's Iteration for Computations with Structured Matrices and Applications, *Computers & Mathematics (with Applications)*, **24**, **3**, 61–75, 1992.

[P92b] V. Y. Pan, Can we utilize the cancelation of the most significant digits? *Tech. Report TR-92-061*, The International Computer Science Institute, Berkeley, 1992

[P93] V. Y. Pan, Decreasing the Displacement Rank of a Matrix, *SIAM J. Matrix Anal. Appl.*, **14**, **1**, 118–121, 1993.

[P93a] V. Y. Pan, Concurrent Iterative Algorithm for Toepliz-like Linear Systems, *IEEE Trans. on Parallel and Distributed Systems*, **4**, **5**, 592–600, 1993.

[P00a] V. Y. Pan, New Techniques for the Computation of Linear Recurrence Coefficients, *Finite Fields and Their Applications*, **6**, 43–118, 2000.

[P00b] V. Y. Pan, Parallel Complexity of Computations with General and Toeplitz-like Matrices Filled with Integers and Extensions, *SIAM Journal on Computing*, **30**, 1080–1125, 2000.

[P01a] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[P01] V. Y. Pan, A Homotopic Residual Correction Process, *Proceedings of the Second Conference on Numerical Analysis and Applications* (P. Yalamov, editor), *Lecture Notes in Computer Science*, **1988**, 644–649, Springer, Berlin, 2001.

[Par80] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[PKRCa] V. Y. Pan, M. Kunin, R. E. Rosholt, H. Cebecioğlu, Residual Correction Algorithms for General and Structured Matrices, to appear.

[PBRZ99] V. Y. Pan, S. Branham, R. Rosholt, A. Zheng, Newton's Iteration for Structured Matrices and Linear Systems of Equations, *Fast Reliable Algorithms for Matrices with Structure*, (T. Kailath, A.H. Sayed, Editors) 189–210, SIAM Publications, Philadelphia, 1999.

[PR01] V. Y. Pan, Y. Rami, Newton's Iteration for the Inversion of Structured Matrices, *Advances in the Theory of Computational Mathematics, Vol. 4: Structured Matrices: Recent Developments in Theory and Computation*, edited by D. A. Bini, E. Tyrtyshnikov and P. Yalamov, 79–90, Nova Science Publishers, Huntington, New York, 2001.

[PRW01] V. Y. Pan, Y. Rami, X. Wang, Structured Matrices and Newton's Iteration: Unified Approach, *Linear Algebra and Its Applications 2001* (in press).

[PS91] V. Y. Pan, R. Schreiber, An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications, *SIAM J. on Scientific and Statistical Computing*, **12**, **5**, 1109–1131, 1991.

[PWa] V. Y. Pan, X. Wang, Inversion of Displacement Operators, to appear.

[PWb] V. Y. Pan, X. Wang, The Norms of the Inverse Displacement Operators, to appear.

[PZHD97] V. Y. Pan, A. Zheng, X. Huang, O. Dias, Newton's Iteration for Inversion of Cauchy-like and Other Structured Matrices, *J. of Complexity*, **13**, 108–124, 1997.

[S33] G. Schultz, Iterative Berechnung der Reciproken Matrix, *Z. Angew. Meth. Mech.*, **13**, 57–59, 1933.

[SS74] T. Söderström, W. Stewart, On the Numerical Properties of an Iterative Method for Computing the Moore–Penrose Generalized Inverse, *SIAM J. Numer. Anal.*, **11**, 61–74, 1974.

[VHK99] M. Van Barel, G. Heinig, P. Kravanja, A Stabilized Superfast Solver for Nonsymmetric Toeplitz Systems, Report TW293, *DCS, Katolieke Universiteit Leuven*, Haverlee, Belgium, 1999.