Lecture 3

Fast and Numerically Stable Algorithms for Structured Matrices

*Six lectures on Algorithms, Trinity term 1999.
Copyright ©1999, R. P. Brent.

Abstract

We consider the numerical stability/instability of fast algorithms for solving systems of linear equations or linear least squares problems with a low displacement-rank structure. For example, the matrices involved may be Toeplitz or Hankel.

In particular, we consider algorithms which incorporate pivoting without destroying the structure, such as the GKO algorithm, and describe some recent results by Sweet and Brent, Ming Gu, Michael Stewart and others on the stability of these algorithms.

It is interesting to compare these results with the corresponding stability results for algorithms based on the seminormal equations and for the well known algorithms of Schur/Bareiss and Levinson.

Outline

- Structured matrices
  - Displacement structure
  - Cauchy-like matrices
  - Toeplitz-like matrices
  - Toeplitz ↔ Cauchy
- Partial pivoting algorithms
  - Possible growth of generators
  - Improvements of Gu and Stewart
- Positive definite structured matrices
  - Schur/Bareiss algorithms
  - Comparison with Levinson
  - Generalised Schur algorithm
- Orthogonal factorisation
  - Weak stability
  - The problem of computing Q

Because of shortage of time, I will not consider look-ahead algorithms or iterative algorithms.

Acronyms

BBH = Bojanczyk, Brent & de Hoog.
BBHS = BBH & Sweet.
GKO = Golberg, Kailath & Olshevsky.

Notation

$R$ is a structured matrix,
$T$ is a Toeplitz or Toeplitz-type matrix,
$P$ is a permutation matrix,
$L$ is lower triangular,
$U$ is upper triangular,
$Q$ is orthogonal.

Error Bounds

In error bounds $O_4(\varepsilon)$ means $O(\varepsilon f(n))$, where $f(n)$ is a polynomial in $n$. 
Stability
Consider algorithms for solving a nonsingular, 
$n \times n$ linear system $Ax = b$.
There are many definitions of numerical stability in the literature. Our definitions follow
those of Bunch [11]. Definition 1 says that the
computed solution has to be the exact solution
of a problem which is close to the original
problem. This is the classical backward stability
of Wilkinson.

Definition 1 An algorithm for solving linear
equations is stable for a class of matrices $A$ if
for each $A$ in $A$ and for each $b$ the computed
solution $\hat{x}$ to $Ax = \hat{b}$ satisfies $\hat{A}\hat{x} = \hat{b}$, where $\hat{A}$
is close to $A$ and $\hat{b}$ is close to $b$.

Note that the matrix $\hat{A}$ does not have to be in
the class $A$. For example, $A$ might be the class
of nonsingular Toeplitz matrices, but $\hat{A}$ need
not be a Toeplitz matrix. (If we do require
$\hat{A} \in A$ we get what Bunch calls strong stability.)

Closeness
In Definition 1, “close” means close in a relative
sense, using some norm, i.e.

$$\|\hat{A} - A\|/\|A\| = O(\varepsilon), \|\hat{b} - b\|/\|b\| = O(\varepsilon).$$

Recall our convention that polynomials in $n$
may be omitted from $O(\varepsilon)$ terms.

We are ruling out faster than polynomial growth
in $n$, such as $O(2^n \varepsilon)$ or $O(n^{2+\varepsilon})$. Perhaps this
is too strict (consider Gaussian elimination).

The Residual
The condition of Definition 1 is equivalent to
saying that the scaled residual

$$\|A\hat{x} - \hat{b}\|/(\|A\| \cdot \|\hat{x}\|)$$
is small.

How Good is the Solution?
Provided $\kappa \varepsilon$ is sufficiently small, stability
implies that

$$\|\hat{x} - x\|/\|x\| = O(\kappa \varepsilon).$$

Weak Stability
Definition 2 An algorithm for solving linear
equations is weakly stable for a class of
matrices $A$ if for each well-conditioned $A$ in $A$
and for each $b$ the computed solution $\hat{x}$ to
$Ax = b$ is such that $\|\hat{x} - x\|/\|x\|$ is small.

In Definition 2, “small” means $O(\varepsilon)$, and
“well-conditioned” means that $\kappa(A)$ is bounded
by a polynomial in $n$. It is easy to see that
stability implies weak stability.
Define the residual

$$r = A\hat{x} - \hat{b}.$$ 

It is well-known that

$$\frac{1}{\kappa} \|b\| \leq \|\hat{x} - x\|/\|x\| \leq \|r\|/\|b\|.$$ 

Thus, for well-conditioned $A$, 

$$\|\hat{x} - x\|/\|x\|$$
is small if and only if $\|r\|/\|b\|$ is small. (This gives an equivalent definition of
weak stability.)

Displacement Structure
Structured matrices $R$ satisfy a Sylvester
equation which has the form

$$\nabla_{\{A_f , A_b\}}(R) = A_f R - RA_b = \Phi \Psi^T,$$

where $A_f$ and $A_b$ have some simple structure
(usually banded, with 3 or fewer full diagonals),
$\Phi$ and $\Psi$ are $n \times \alpha$ and $\alpha \times n$ respectively, and $
\alpha$ is some (small) integer.

The pair of matrices $(\Phi, \Psi)$ is called the
$\{A_f , A_b\}$-generator of $R$.

$\alpha$ is called the $\{A_f , A_b\}$-displacement rank of $R$.
We are interested in cases where $\alpha$ is small (say
at most 4).
Example – Cauchy

Particular choices of $A_f$ and $A_b$ lead to definitions of basic classes of matrices. Thus, for a Cauchy matrix

$$C(t,s) = \begin{bmatrix} 1 & \ldots & 0 \\ t_1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ t_m & \ldots & s_j \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 1 \end{bmatrix},$$

we have

$$A_f = D_1 = \text{diag}(t_1,t_2,\ldots,t_n),$$

$$A_b = D_s = \text{diag}(s_1,s_2,\ldots,s_n)$$

and

$$\Phi^T = \Psi = [1,1,\ldots,1].$$

More general matrices, where $\Phi$ and $\Psi$ are any rank-$\alpha$ matrices, are called Cauchy-type.

Example – Toeplitz

For a Toeplitz matrix $T = [t_{ij}] = [a_{i-j}]$

$$A_f = Z_1 = \begin{bmatrix} 0 & 1 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{bmatrix},$$

$$A_b = Z_{-1} = \begin{bmatrix} 0 & 1 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 \end{bmatrix},$$

$$\Phi = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ a_0 & a_1 & \ldots & a_{n-1} + a_n-1 \end{bmatrix}^T$$

and

$$\Psi = \begin{bmatrix} a_{n-1} - a_{n-2} & \ldots & a_2 - a_1 & a_1 & a_0 \\ 0 & \ldots & 0 & 1 \end{bmatrix}.$$
Theorem 1  Let a matrix $R_1 = [d_1 \ W_1^T; 
 y_1 \ R_1]$ satisfy the Sylvester equation

\[ \nabla_{X_{11}} (R_1) = A_{11} R_1 - R_1 A_{11} = \Phi^{(1)} \Psi^{(1)}, \]

where $\Phi^{(1)} = [\varphi_{1}^{(1)}T \ \varphi_{2}^{(1)}T \ \cdots \ \varphi_{n}^{(1)}T]^T$, $\Psi^{(1)} = [\psi_{1}^{(1)} \ \psi_{2}^{(1)} \ \cdots \ \psi_{n}^{(1)}]$, $\varphi_i^{(1)} \in \mathbb{C}^{l \times n}$ and $\psi_i^{(1)} \in \mathbb{C}^{n \times n}$, ($i = 1, 2, \ldots, n$). Then $R_2$, the Schur complement of $d_1$ in $R_1$, satisfies the Sylvester equation

\[ \nabla_{X_{22}} (R_2) = A_{22} R_2 - R_2 A_{22} = \Phi^{(2)} \Psi^{(2)}, \]

where $A_{22}$ and $A_{11}$ are respectively $A_{11}$ and $A_{11}$ with their first rows and first columns deleted, and where

\[ \Phi^{(2)} = [0, \varphi_{2}^{(2)}T, \varphi_{3}^{(2)}T, \cdots, \varphi_{n}^{(2)}T]^T \]

and

\[ \Psi^{(2)} = [0, \psi_{2}^{(2)} \psi_{3}^{(2)} \cdots \psi_{n}^{(2)}] \]

are given by

\[ \Phi^{(2)}_{[2:n]} = \Phi^{(1)}_{[2:n]} - y_1 \psi_1^{(1)}/d_1, \]

\[ \psi_1^{(2)}_{[2:n]} = \psi_1^{(1)} - y_1 \Psi^{(1)}_{[1:2]} / d_1. \]

Structured Gaussian elimination

Algorithm 1 (Structured Gaussian elimination)

1. Recover from the generator $\Phi^{(1)}, \Psi^{(1)}$ the first row and column of

\[ R_1 = \begin{bmatrix} d_1 & W_1^T \\ y_1 & R_1 \end{bmatrix}. \]

2. $[y_1^T / d_1^T]$ and $[d_1^T W_1^T]$ are respectively the first column and row of $L_1$ and $U_1$ in the LU factorisation of $R_1$.

3. Compute by equations (1) and (2), the generator $(\Phi^{(2)}, \Psi^{(2)})$ for the Schur complement of $d_1$ in $R_1$.

4. Proceed recursively with $\Phi^{(2)}$ and $\Psi^{(2)}$. Each major step yields $[y_k^T / d_k]^T$ and $[d_k^T W_k^T]$, which are respectively the first column and row of $L_k$ and $U_k$ in the LU factorisation of $R_k$. Column $k$ of $L$ and row $k$ of $U$ are respectively $[0_k^T 
 1 \ y_k^T / d_k]^T$ and $[0_k^T 
 1 \ d_k^T \ W_k^T]$.

Partial Pivoting

Row and/or column interchanges destroy the structure of matrices such as Toeplitz matrices. However, if $A_f$ is diagonal (which is the case for Cauchy and Vandermonde type matrices), then

the structure is preserved under row permutations.

This observation leads to the GKO-Cauchy algorithm for fast factorisation of Cauchy-type matrices with partial pivoting, and many recent variations on the theme by Boros, Golberg, Ming Gu, Heining, Kailath, Olhevsky, M. Stewart, et al.

Toeplitz to Cauchy

Heining (1994) showed that, if $T$ is a Toeplitz-type matrix, then

\[ R = F T D^{-1} F^* \]

is a Cauchy-type matrix, where

\[ F = \frac{1}{\sqrt{n}} [e^{2\pi i (k-1)(j-1)/n}]_{1 \leq k, j \leq n} \]

is the Discrete Fourier Transform matrix,

\[ D = \text{diag}(1, e^{\pi i/n}, \ldots, e^{\pi i(n-1)/n}), \]

and the generators of $T$ and $R$ are related by unitary transformations (see [38, Thm. 2.2] for the details).

The transformation $T \leftrightarrow R$ is perfectly stable because $F$ and $D$ are unitary.

Note that $F$ and $R$ are (in general) complex even if $T$ is real.
GKO-Toeplitz
As pointed out by Heinig (1994) and exploited by GKO (1995), it is possible to convert the
generators of $T$ to the generators of $R$ in
$O(n \log n)$ operations via FFTs. $R$ can then be
factorised as $R = P^T LU$ using GKO-Cauchy.
Thus, from the factorisation

$$T = F^* P^T LU FD,$$

a linear system involving $T$ can be solved in
$O(n^2)$ (complex) operations.
Other structured matrices, such as
Toeplitz-plus-Hankel, Vandermonde,
Chebyshev-Vandermonde, etc, can be converted
to Cauchy-type matrices in a similar way.

Error Analysis
Because GKO-Cauchy (and GKO-Toeplitz)
involve partial pivoting, we might guess that
their stability would be similar to that of
Gaussian elimination with partial pivoting.

The Catch
Unfortunately, there is a flaw in the above
reasoning. During GKO-Cauchy the generators
have to be transformed, and the partial pivoting
does not ensure that the transformed generators
are small.
Sweet & Brent (1995) show that significant
generator growth can occur if all the elements of
$\Phi \Psi$ are small compared to those of $[\Phi][\Psi]$. This
can not happen for ordinary Cauchy matrices
because $\Phi^{(k)}$ and $\Psi^{(k)}$ have only one column
and one row respectively. However, it can
happen for higher displacement-rank
Cauchy-type matrices, even if the original
matrix is well-conditioned.

The Toeplitz Case
In the Toeplitz case there is an extra constraint
on the selection of $\Phi$ and $\Psi$, but it is still
possible to give examples where the normalised
solution error grows like $\kappa^2$ and the normalised
residual grows like $\kappa$, where $\kappa$ is the condition
number of the Toeplitz matrix. Thus, the
GKO-Toeplitz algorithm is (at best) weakly
stable.
It is easy to think of modified algorithms which
avoid the examples given by Sweet & Brent, but
it is difficult to prove that they are stable in all
cases. Stability depends on the worst case,
which may be rare and hard to find by random
sampling.

Gu and Stewart’s Improvements
The problem with the original GKO algorithm
is growth in the generators. Ming Gu suggested
exploiting the fact that the generators are not
unique.
Recall the Sylvester equation

$$\nabla_{[A_f, A_b]}(R) = A_f R - R A_b = \Phi \Psi,$$

where the generators $\Phi$ and $\Psi$ are $n \times \alpha$ and
$\alpha \times n$ respectively. Clearly we can replace $\Phi$ by
$\Phi M$ and $\Psi$ by $M^{-1} \Psi$, where $M$ is any
invertible $\alpha \times \alpha$ matrix, because this does not
change the product $\Phi \Psi$. Similarly at later
stages of the GKO algorithm.
Ming Gu (1995) proposes taking $M$ to
orthogonalize the columns of $\Phi$ (that is, at each
stage we do an orthogonal factorisation of the
generators). Michael Stewart (1997) proposes a
(cheaper) LU factorisation of the generators. In
both cases, clever pivoting schemes give error
bounds analogous to those for Gaussian
elimination with partial pivoting.
Gu and Stewart’s Error Bounds

The error bounds obtained by Ming Gu and Michael Stewart involve an exponentially growing factor $K^n$ where $K$ depends on the ratio of the largest to smallest modulus elements in the Cauchy matrix

$$\begin{bmatrix}
  1 \\
  t_i - s_j 
\end{bmatrix}.$$ 

Although this is unsatisfactory, it is similar to the factor $2^{n-1}$ in the error bound for Gaussian elimination with partial pivoting.

Michael Stewart (1997) gives some interesting numerical results which indicate that his scheme works well, but more numerical experience is necessary before a definite conclusion can be reached.

In practice, we can use an $O(n^2)$ algorithm such as Michael Stewart’s, check the residual, and resort to iterative refinement or a stable $O(n^3)$ algorithm in the (rare) cases that it is necessary.

Bareiss – Positive Definite Case

BBHS (1995) have shown that the numerical properties of the Bareiss algorithm are similar to those of Gaussian elimination (without pivoting). Thus, the algorithm is stable for positive definite symmetric $T$.

The Levinson algorithm can be shown to be weakly stable for bounded $n$, and numerical results by Varah, BBHS and others suggest that this is all that we can expect. Thus, the Bareiss algorithm is (generally) better numerically than the Levinson algorithm.

Cybenko showed that if certain quantities called “reflection coefficients” are positive then the Levinson-Durbin algorithm for solving the Yule-Walker equations (a positive-definite system with special right-hand side) is stable. However, “random” positive-definite Toeplitz matrices do not usually satisfy Cybenko’s condition.

Positive Definite Structured Matrices

An important class of algorithms, typified by the algorithm of Bareiss (1969), find an $LU$ factorisation of a Toeplitz matrix $T$, and (in the symmetric case) are related to the classical algorithm of Schur for the continued fraction representation of a holomorphic function in the unit disk.

It is interesting to consider the numerical properties of these algorithms and compare with the numerical properties of the Levinson\textsuperscript{1} algorithm (which essentially finds an $LU$ factorisation of $T^{-1}$).

\textsuperscript{1}Discovered independently by Kolmogorov and Wiener in 1941.

The Generalised Schur Algorithm

The Schur algorithm can be generalised to factor a large variety of structured matrices – see Kailath and Chua (1994) or Kailath and Sayed (1995). For example, the generalised Schur algorithm applies to block Toeplitz matrices, Toeplitz block matrices, and to matrices of the form $T^T T$ where $T$ is rectangular Toeplitz.

It is natural to ask if the stability results of BBHS (which are for the classical Schur/Bareiss algorithm) extend to the generalised Schur algorithm. This was considered by M. Stewart and Van Dooren (1997), and also (in more generality) by Chandrasekharan and Sayed (1998).

The conclusion is that the generalised Schur algorithm is stable for positive definite matrices, provided that the hyperbolic transformations in the algorithm are implemented correctly. In contrast, BBHS showed that stability of the classical Schur/Bareiss algorithm is not so dependent on details of the implementation.
Fast Orthogonal Factorisation

In an attempt to achieve stability without pivoting, and to solve \( m \times n \) least squares problems, it is natural to consider algorithms for computing an orthogonal factorisation

\[ T = QU \]

of \( T \). The first such \( O(n^3) \) algorithm\(^2\) was introduced by Sweet (1982–84). Unfortunately, Sweet’s algorithm is unstable.

Other \( O(n^3) \) algorithms for computing the matrices \( Q \) and \( U \) or \( U^{-1} \) were given by BBH (1986), Chun et al (1987), Cybenko (1987), and Qiao (1988), but none of them has been shown to be stable, and in several cases examples show that they are unstable.

\(^2\)For simplicity the time bounds assume \( m = O(n) \).

3–25

The Problem – \( Q \)

Unlike the classical \( O(n^3) \) Givens or Householder algorithms, the \( O(n^2) \) algorithms do not form \( Q \) in a numerically stable manner as a product of matrices which are (close to) orthogonal.

For example, the algorithms of Bojanczyk, Brent and de Hoog (1986) and Chun et al (1987) depend on Cholesky downdating, and numerical experiments show that they do not give a \( Q \) which is close to orthogonal.

The generalised Schur algorithm, applied to \( T^T T \), computes the upper triangular matrix \( U \) but not the orthogonal matrix \( Q \).

3–26

The Saving Grace – \( U \) and Semi-Normal Equations

It can be shown (BBH, 1995) that, provided the Cholesky downdates are implemented in a certain way (analogous to the condition for the stability of the generalised Schur algorithm) the BBH algorithm computes \( U \) in a weakly stable manner. In fact, the computed upper triangular matrix \( \tilde{U} \) is about as good as can be obtained by performing a Cholesky factorisation of \( T^T T \), so

\[ \|T^T T - \tilde{U}^T \tilde{U}\| \leq O(n^2). \]

Thus, by solving

\[ \tilde{U}^T \tilde{U} x = T^T b \]

(the so-called semi-normal equations) we have a weakly stable algorithm for the solution of general Toeplitz systems \( T x = b \) in \( O(n^2) \) operations. The solution can be improved by iterative refinement if desired.

Note that the computation of \( Q \) is avoided.

3–27

Computing \( Q \) Stably

It is difficult to give a satisfactory \( O(n^2) \) algorithm for the computation of \( Q \) in the factorisation

\[ T = QU \]

Chandrasekharan and Sayed get close – they give a stable algorithm to compute the factorisation

\[ T = LQU \]

where \( L \) is lower triangular, provided that \( T \) is square. Their algorithm can be used to solve linear equations but not for the least squares problem. Also, because their algorithm involves embedding the \( n \times n \) matrix \( T \) in a \( 2n \times 2n \) matrix

\[
\begin{bmatrix}
T^T T & T^T \\
T & 0
\end{bmatrix},
\]

the constant factors in the operation count are large: \( 59n^2 + O(n \log n) \), compared to \( 8n^2 + O(n \log n) \) for BBH and seminormal equations.

3–28
Some Open Questions

- How do the GKO and similar algorithms using partial pivoting compare with the “lookahead” algorithms of Chan and Hansen [13], Freund and Zha [19], Gutknecht [25], and others?
- Is there a stable (not just weakly stable) fast algorithm for the (rectangular) structured least squares problem?
- What can be said about the stability (or instability) of the “superfast” algorithms whose running time is
  \[ O\left(n(\log n)^2\right) \]?

For these algorithms see Ammar and Gragg [1], Brent, Gustavson and Yun [9].

- What are the best generalisations to block-structured problems, e.g. block Toeplitz with \(\sqrt{n} \times \sqrt{n}\) blocks?

References


