Un'applicazione dell'algoritmo GKO e problemi collegati

A. Aricò and G. Rodriguez

Dipartimento di Matematica e Informatica
Università di Cagliari

Due giorni di Algebra Lineare Numerica
Bologna, 6-7 marzo 2008

Introduction

Many techniques has been devised to solve Toeplitz linear systems

$$T_n x = b$$

...but, if you look for a reliable solver that you can use in ©MATLAB, it seems that the choices are not so many.

This talk does not claim *new* results about Toeplitz systems: we just used some classical tools, namely:

- matrix representation via displacement structure
- Schur algorithm

in order to write a *simple* code which works under *general* assumptions. Roughly speaking, such a code should work as

```
[ x, rcond ] = tsolve( col1, row1, rhs );
```

and give a *solution* x whenever T_n is nonsingular.

Outline

Toeplitz representation via displacement structure

$$n, m \in \mathbb{N}, \quad t_{1-m}, \dots, t_{n-1} \in \mathbb{C}, \quad T_{n,m} = [t_{i-j}]_{\substack{i=1..n\\ j=1..m}}, \quad Z_n(\phi) = \left[\begin{array}{c|c} & \phi \\ \hline I_{n-1} & \end{array}\right]$$

Then it's well known that

$$\underbrace{Z_n(\phi)\,T_{n,m} - T_{n,m}Z_m(\psi)}_{\text{displacement of } T_{n,m}} = \begin{bmatrix} * & * & * & * \\ & & & * \\ & & & * \end{bmatrix} = \underbrace{G\ H^T}_{\text{generators}}$$

where $G = [\mathbf{a}, \mathbf{e}_1]$ is $n \times 2$, $H = [\mathbf{e}_m, \mathbf{b}]$ is $m \times 2$ and, with a little abuse,

$$\mathbf{a} = [\psi\, t_{n-i} - t_{-i}]_{i=1}^{m} \qquad \mathbf{b} = [t_{i-m} - \psi\, t_i]_{i=0}^{n-1}.$$

This is a $T_{n,m}$ representation by displacement structure.
In the square case with $\phi = -\psi = 1$ we have

$$Z_n(1)\,T_n - T_n Z_n(-1) = \left[\begin{array}{c|c} t_0 & 1 \\ \hline t_1 + t_{1-n} & \\ \dots & \\ t_{n-1} + t_{-1} & \end{array}\right] \left[\begin{array}{c|c} & t_{n-1} - t_{-1} \\ & \dots \\ & t_1 - t_{1-n} \\ \hline 1 & t_0 \end{array}\right]^T$$

From Toeplitz-like to Cauchy-like

We change the displacement representation of T_n in a similar one, which is more suitable to apply row exchange

- n-roots of 1 and -1: $\omega_n^+(j) = \mathrm{e}^{\mathbf{i}\frac{\pi}{n}(2j)}$ and $\omega_n^-(j) = \mathrm{e}^{\mathbf{i}\frac{\pi}{n}(2j+1)}$
- unitary matrices: $D_n^\pm = \operatorname*{diag}_{j=0..n-1}\left(\omega_n^\pm(j)\right)$ and $F_n^\pm = \frac{1}{\sqrt{n}}\left[\left(\omega_n^\pm(i)\right)^j\right]_{i,j=0\ldots n-1}$
- diagonalization: $Z_n(\pm 1) = (F_n^\pm)^* \, D_n^\pm \, F_n^\pm$

Then

$$Z_n(1)\,T_n - T_n Z_n(-1) = G\,H^* \qquad \text{(a)}$$

$$\Downarrow$$

$$D_n^+ \, \underbrace{F_n^+ \, T_n (F_n^-)^*}_{} - \underbrace{F_n^+ \, T_n (F_n^-)^*}_{} \, D_n^- = (F_n^+ G)\,(F_n^- H)^* \qquad \text{(b)}$$

$$\Downarrow$$

$$D_n^+ \, C_n - C_n \, D_n^- = G_C \, H_C^* \qquad \text{(c)}$$

and, since $D_n^\pm$ are diagonal, C_n is a *Cauchy-like* matrix (Heinig, GKO).
Computational cost: two `FFT` on **a** and **b** to get G_C and H_C.
Then solve $C_n\,(F_n^- \mathbf{x}) = F_n^+ \mathbf{b}$. ... don't compute C_n!

Solving the Cauchy-like system

We resorted to a matrix C_n implicitely defined by

$$D_n^+ C_n - C_n D_n^- = G_C H_C^*$$

and we need to solve $C\,(F_n^- \mathbf{x}) = F_n^+ \mathbf{b}$. We apply the Schur algorithm to

$$\widetilde{C}_n = \begin{bmatrix} C_n & F_n^+ \mathbf{b} \\ -I_n & O \end{bmatrix}_{(2n)\times(n+1)}$$

and get $(\widetilde{C}_n / C_n) = C_n^{-1} F_n^+ \mathbf{b}$; then $\mathbf{x} = (F_n^-)^* \, (\widetilde{C}_n / C_n)$. Note that

$$\begin{bmatrix} D_n^+ & \\ & D_n^- \end{bmatrix} \widetilde{C}_n - \widetilde{C}_n \begin{bmatrix} D_n^- & \\ & O_{n\times 1} \end{bmatrix} = \underbrace{\begin{bmatrix} G_C & F_n^+ Z_n^+ \mathbf{b} \\ O_{n\times 2} & O_{n\times 1} \end{bmatrix}}_{\widetilde{G}} \underbrace{\begin{bmatrix} H_C & O_{n\times 1} \\ O_{1\times n} & I_1 \end{bmatrix}^*}_{\widetilde{H}^*} \qquad \text{(a)}$$

so the truncated Schur algorithm (GKO) can be applied:

- cost to compute $\widetilde{G}$ and $\widetilde{H}$: 3 FFT $= O(n \log(n))$
- cost to apply Schur (to $\widetilde{C}_n$, implicitely via (a)): $O(n^2)$.

Some considerations

- we perform partial pivoting
- pivoting restricted to row=1...n $\quad\Rightarrow\quad (\widetilde{C}/C)$ "is the solution"
- the subroutine `clsolve`
  - is written in `C` (+ `mex`)
  - let us extimate $\mu_1(T)$

$$\Pi A = LU$$

$$\Downarrow$$

$$\begin{bmatrix} \Pi & \\ & I \end{bmatrix} \begin{bmatrix} A & B \\ -I & O \end{bmatrix} = \begin{bmatrix} L & \\ -U^{-1} & I \end{bmatrix} \begin{bmatrix} U & L^{-1}\Pi B \\ & A^{-1}B \end{bmatrix}$$

U by rows and U^{-1} by cols

The routines

```
>> help drsolve
MATLAB Displacement Rank Structure Solver (DRSOLVE)

General.
t2tl       -   Conversion from Toeplitz to Toeplitz-like
tl2cl      -   Conversion from Toeplitz-like to Cauchy-like
t2cl       -   Conversion from Toeplitz to Cauchy-like
cl2full    -   Cauchy-like matrix from its generators
tsolve     -   Toeplitz system solver (based on GKO)
clsolve    -   Cauchy-like system solver (based on GKO)
fouriermv  -   Fourier Matrix-Vector product
nroots1    -   n-th roots of |z|=1
```

```
[ x, rcond ] = tsolve( col1, row1, rhs );
```

TOMS729

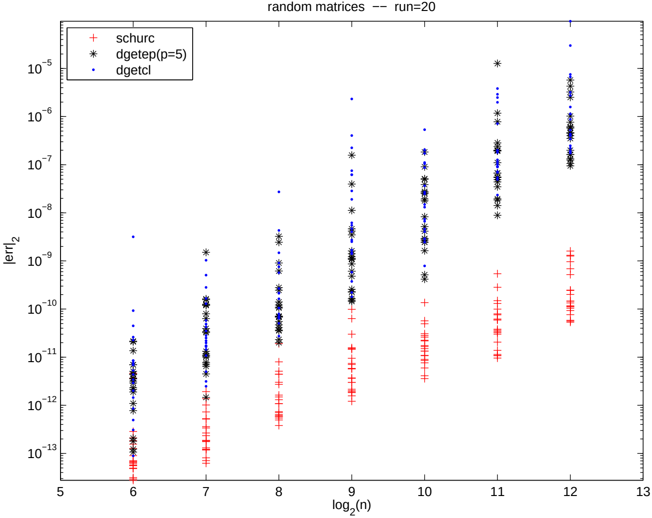
We compared our code with the routines `dgetcl` and `dgetep`, by T. Chan and P.C. Hansen (TOMS729):

- that code is written in FORTRAN
- algorithm is Levinson (`cl`) and extended($p = 5$) Levinson (`ep`)
- both works on double precision real data
- we did not use the simmetric version (`dsyt**`)

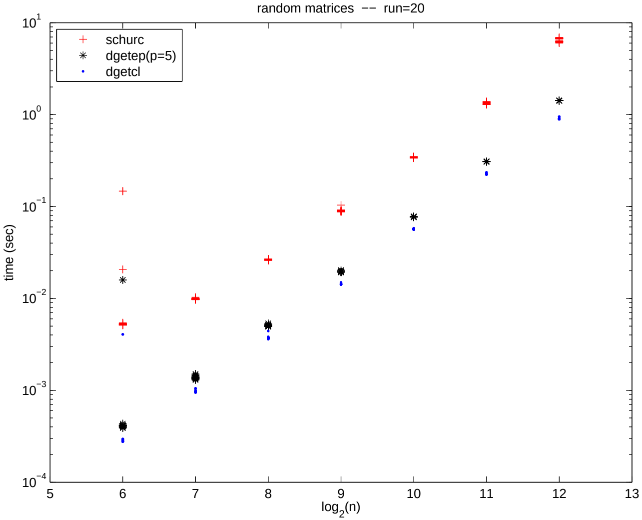
We applied `tsolve`, `dgetcl` and `dgetep` to three class of marices

- random with $n = 2^t$, $t = 6, \ldots, 12$ (20 times)
- KMS($\rho = 0.5$), $t = 6, \ldots, 12$ (5 times)
- KMS($\rho = 0.999$), $t = 6, \ldots, 12$ (5 times)

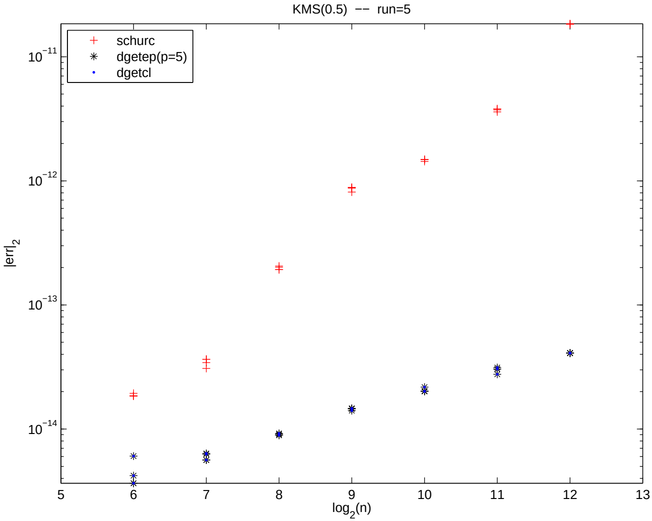
Random matrices



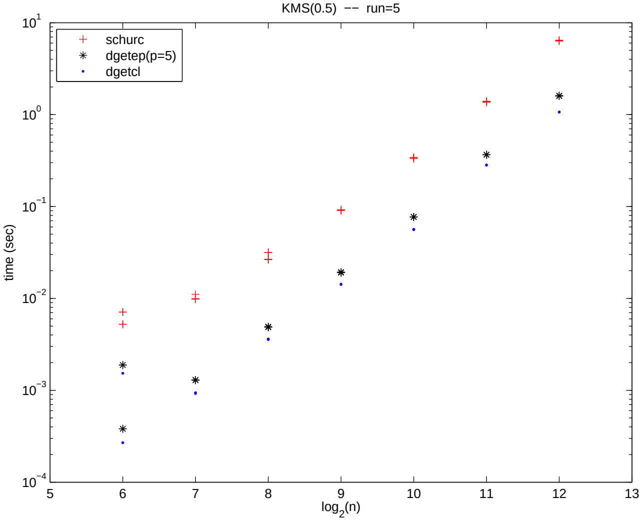
Random matrices



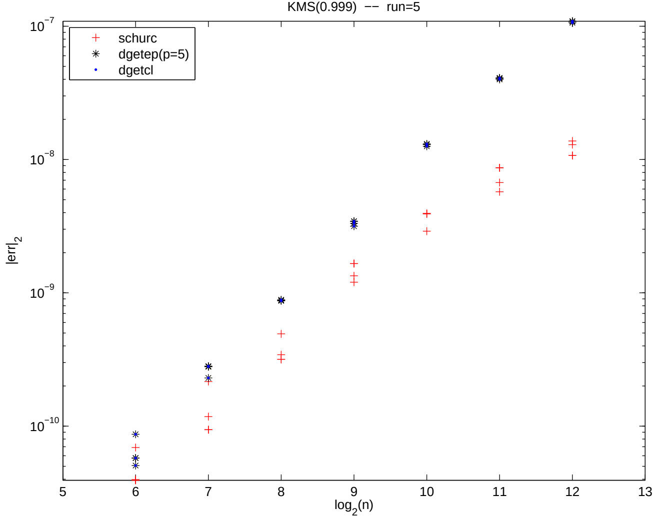
KMS matrices



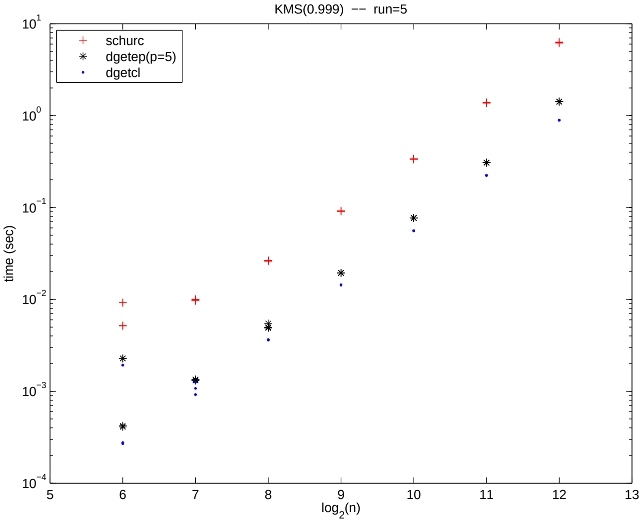
KMS matrices



KMS matrices



KMS matrices



Conclusion and Open problems

Conclusion

- we presented a solver which has been implemented in ©MATLAB
- this code is available,

```
[ x, rcond ] = tsolve( col1, row1, rhs );
```

- its computational cost is $O(n^2)$
- it works under general assumptions, i.e. T_n nonsingular
- a similar code has been written for the least square problem

Open problems

- several kinds of pivoting have been proposed (...)
  - classical
  - ad hoc (Sweet & Brent, M. Gu, ...)
- scaling of the generators?
- apply a similar idea to the rank-deficient least square problem