Risolv - Robust Iterative Solver

Hillel Tal-Ezer

Academic College of Tel-Aviv Yaffo

hillel@mta.ac.il

Abstract

One of the most effective iterative algorithms for solving large, sparse, non-symmetric linear system is GMRES. Nevertheless, it suffers from lack of robustness. The algorithm can exhibit very slow rate of convergence or complete stagnation. In this talk we would like to present a new iterative algorithm, named Risolv, aiming at overcoming this drawback. The algorithm is based on using Leja points for constructing the 'optimal' polynomial $P_m(z)$ which satisfies

$$||P_m(z)||_\infty = \min_{Q_m \in \Pi_m} ||Q_m(z)||_\infty \qquad z \in D \tag{1}$$

where Π_m is the set of all polynomials of degree m which satisfies $Q_m(0) = 1$ and D is a domain in the complex plane which includes all the eigenvalues of A. The new algorithm is more robust and specially efficient in the case where we have a set of linear systems which share the same matrix A and differ by the r.h.s vector.

In this talk we will address the essential problem of solving large set of linear equations

$$Ax = b \tag{2}$$

where A is square, general, non-symmetric $N \times N$ matrix.

When A is symmetric, positive definite , Conjugate Gradient is the way to go.

There are optimal algorithms for the indefinite, symmetric case (e.g. Minres, Symmlq).

The situation is significantly more complicated in the general, non-symmetric case.

A popular algorithm for this type of problems is Gmres .

Gmres is optimal in the following sense:

Let x_m be the solution vector after applying m matrix-vector multiplications and r_m is the residum vector

$$r_m = b - Ax_m, \tag{3}$$

it can be shown that

$$r_m = P_m(A)r_0 \tag{4}$$

where $P_m(z)$ is a polynomial of degree m which satisfies $P_m(0) = 1$ and r_m satisfies

$$||r_m||_2 = \min_u ||b - Au||_2. \tag{5}$$

Disadvantage - one has to store m vectors. m can be very large.
Solution - restarted Gmres $\rightarrow$ GMRES (k) $\rightarrow$ keep in memory only $k + 1$ Krylov vectors.
Restarted Gmres leads to loss of optimality. Can result in extremely slow rate of convergence or complete stagnation.
Another approach is based on polynomial approximation in the complex plane.

PAC - Polynomial Approximation in the Complex Plane

We have (4)

$$r_m = P_m(A) r_0. \tag{6}$$

Let $u_i, \lambda_i,\ 1 \le i \le N$ be eigenvectors and corresponding eigenvalues of A. Assuming that the set of eigenvectors is linearly independent, we can write

$$r_0 = \sum_{i=1}^{N} \beta_i u_i \tag{7}$$

and

$$r_m = \sum_{i=1}^{N} \beta_i P_m(\lambda_i) u_i. \tag{8}$$

In the PAC approach, the algorithm is based on looking for P_m such that $\max_{z \in D} |P_m(z)|$ is as small as possible (D is the domain in the complex plane which includes all the eigenvalues of A).

Examples:

Examples:

- Chebyshev polynomials when D is real interval or elipse in the complex plane.
- Faber polynomials when D is in the complex plane.
- Leja interpolating polynomials when D is in the complex plane.

drawback- need knowledge of D.

advantages:

- If D does not surround the singular point $z = 0$, convergence is guaranteed.
- There is no need for inner-products. Significant saving of cpu, specially in parallel computations.

The new algorithm presented in this talk belongs to this type of algorithms and it overcomes the drawback mentioned above.

RISOLV

Although Risolv is a PAC algorithm, it can be considered as a slight modification of GMRES.

Using GMRES , at the j restart, we have

$$x_{j+1} = x_j + V_j y_j \tag{9}$$

where y_j is chosen such that $||r_{j+1}||_2$ is minimized.

r_{j+1} satisfies

$$r_{j+1} = \prod_{i=1}^{k} \left(I - \frac{1}{z_{ij}} A \right) r_j \tag{10}$$

where $z_{1j}, \ldots, z_{kj}$ are the eigenvalues of the modified Hessenberg matrix.

Hence, after n restarts we have

$$r_m = \prod_{j=1}^{n} \prod_{i=1}^{k} \left(I - \frac{1}{z_{ij}} A \right) r_0. \qquad (m = nk) \tag{11}$$

Also in the Risolv case, x_{j+1} satisfies (9) but now,

y_j is chosen such that $z_{1j}, \ldots, z_{kj}$ are uniformly distributed in D.

This target can be achieved by making use of Leja algorithm.

Leja Polynomial

Sequences of Leja points , z_j , in D are defined recursively as follows: if z_0 is an arbitrary fixed point in D , the z_j are chosen in such a way that

$$\prod_{k=0}^{j-1} |z_j - z_k| = \max_{z \in D} \prod_{k=0}^{j-1} |z - z_k|, \qquad j = 1, 2, \ldots \tag{12}$$

This set of points can be approximated by

$$\prod_{k=0}^{j-1} |z_j - z_k| = \max_{z \in \hat{D}} \prod_{k=0}^{j-1} |z - z_k|, \qquad j = 1, 2, \ldots \tag{13}$$

where $\hat{D}$ is a large set of points distributed all over D.

Leja points are uniformly distributed in D and if

$$P_m(z) = \prod_{i=0}^{m-1} \left(1 - \frac{z}{z_i}\right) \tag{14}$$

then $\max_{z \in D} |P_m(z)|$ converges to 0 exponentially fast.

Employing Leja algorithm in Risolv

Let H_i be the Hessenberg matrix that satisfies

$$AV = VH_i + h_{i+1,i} v_{i+1} e_i^H, \qquad 1 \leq i \leq k \tag{15}$$

and $\hat{H}_i$ is the modified Hessenberg defined as

$$\hat{H}_i = H_i + h_{i+1,i}^2 f e_i^H, \tag{16}$$

where $f = H_i^{-H} e_i$ and $e_i = [0, \ldots, 1]^H$.

At each restart we have in our possession $2k$ matrices, $H_i, \hat{H}_i$, $1 \leq i \leq k$. Each H_i matrix has i eigenvalues. All together we have a set of k^2 eigenvalues. We consider this set as $\hat{D}$ from which we extract k , uniformly distributed, Leja points (taking into account the already chosen $k \times j$ points).

Hence, after n restarts, the residual is the same as (9) but now the set of points

$$\{z_{ij}\}_{1\leq i\leq k, 1\leq j\leq n} \tag{17}$$

is almost uniformly distributed in D.

Computing y_j and e_j at each cycle

At the $j's$ stage we have

$$x_{j+1} = x_j + Vy_j, \qquad r_{j+1} = \hat{V}e_j \tag{18}$$

where y_j and e_j can be computed by the following:

```
y=zeros(k,1)

e=zeros(k+1,1)

e(1)=||r||

for i=1:k

   w=(1/z(ij))*e(1:k)

   y=y+w

   e=e-h(1:k+1,1:k)*w

end
```

multiple right hand sides

RISOLV is specially efficient while solving

$$Ax^i = b^i, \qquad 1 \leq i \leq s. \tag{19}$$

After solving the first system , the set $\{z_{ij}\}$ is almost uniformly distributed and RISOLV turns to be mainly a simple RICHARDSON algorithm

$$x_{k+1} = x_k + \frac{1}{w_k} \left(b - Ax_j \right). \tag{20}$$

where w_k is a point in the set $\{z_{ij}\}$.

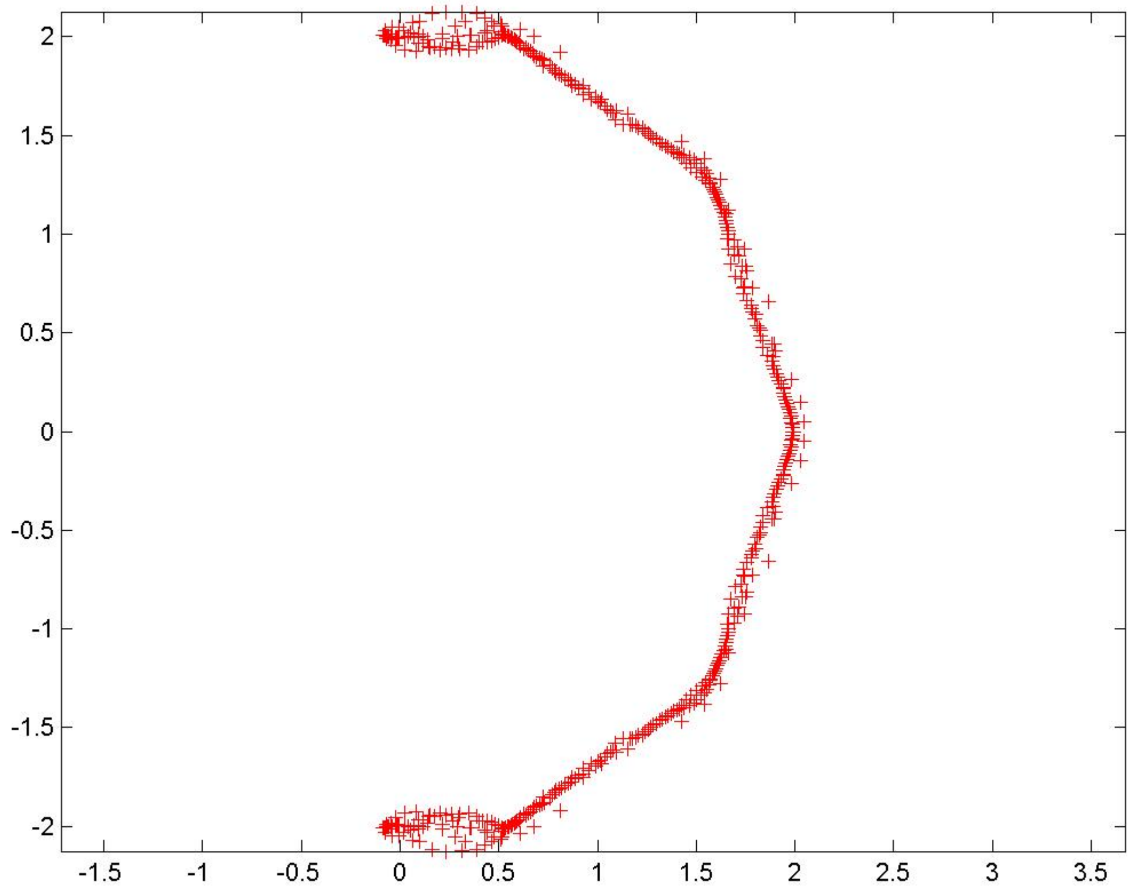
Obviously, this stage is free of inner products. If needed, few additional matrix-vector multiplications will reduce the error to the desired accuracy.

NUMERICAL EXAMPLES

Example 1: Modified Grcar matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & & & \\ -1 & 1 & 1 & 1 & 1 & 1 & 1 & & \\ & -1 & 1 & 1 & 1 & 1 & 1 & 1 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

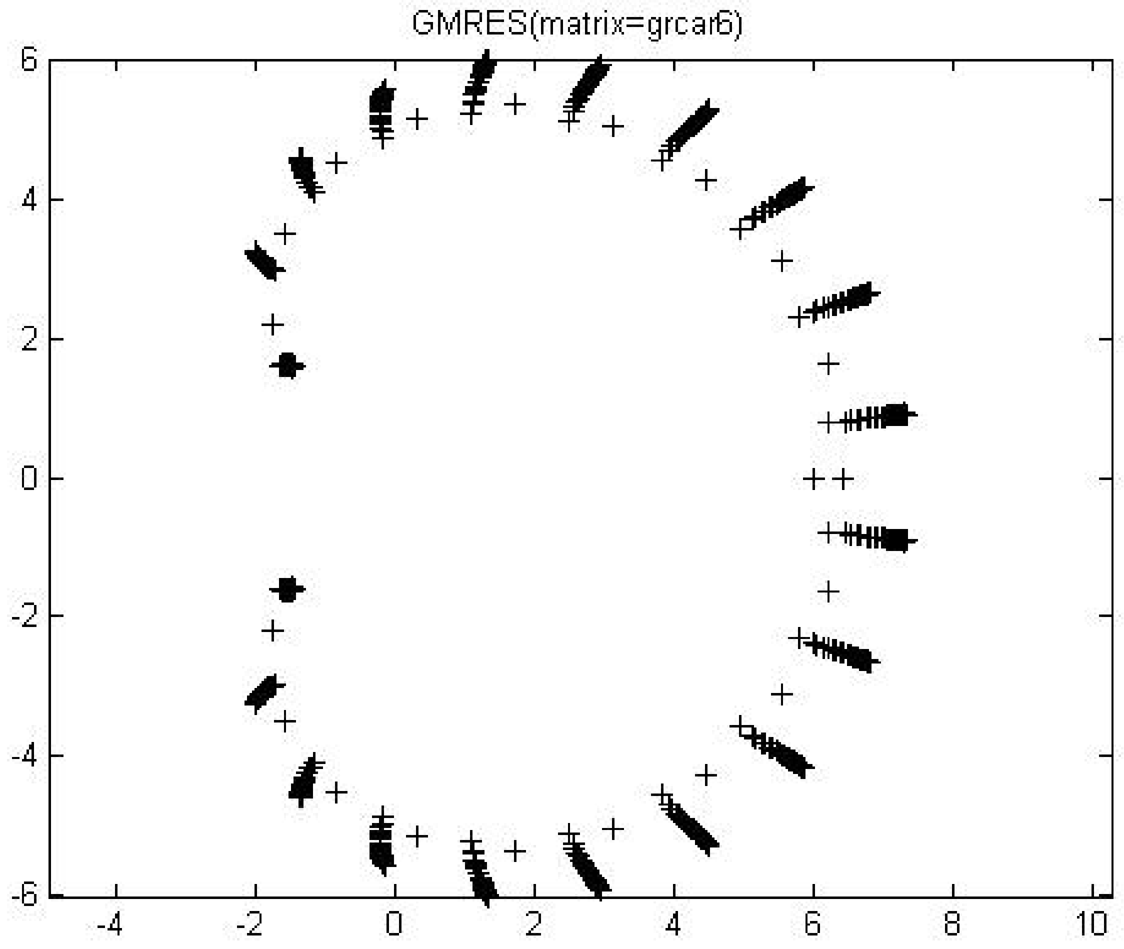
The size of the matrix in this example is 500×500 and $b = [1, \ldots, 1]^T$. The eigenvalues are

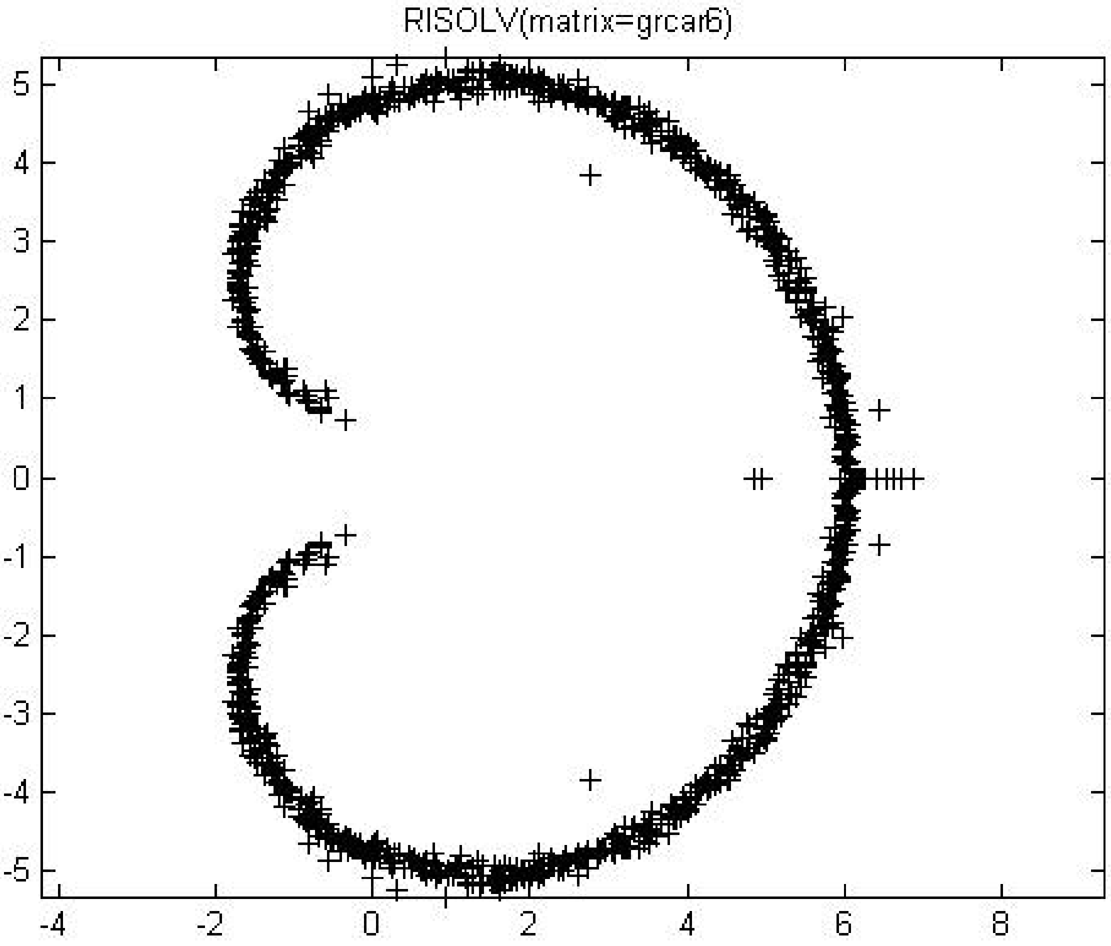


We have solved this system with GMRES and RISOLV. For both of them, the size of the Krylov space was 20 and the desired accuracy was 10^{-4}.

It took GMRES and RISOLV 3226 and 1195 matrix-vector multiplications respectively to reach a solution.

The following figures are of the two sets z_{ij} (11) of Gmres and Risolv.

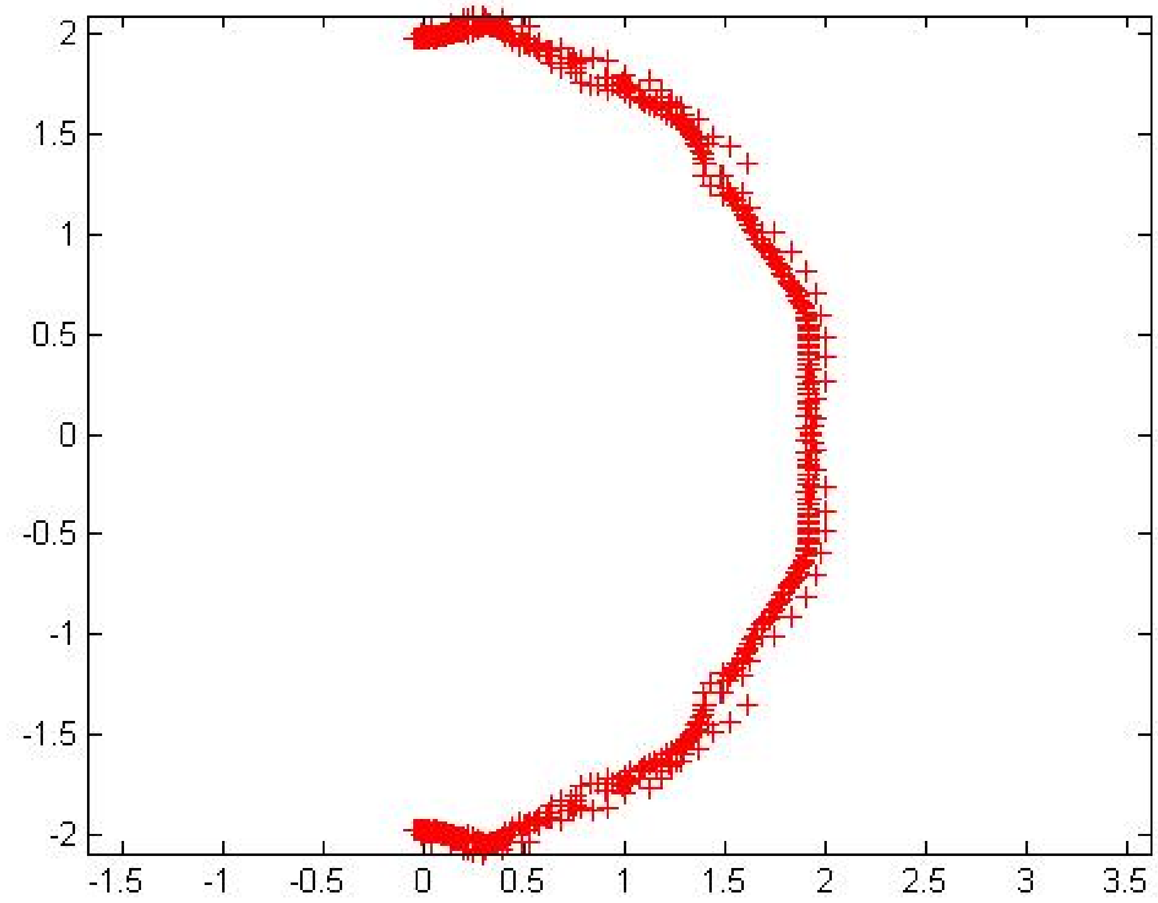

GMRES(matrix=grcar6)
6
4
2
0
-2
-4
-6
-4
-2
0
2
4
6
8
10



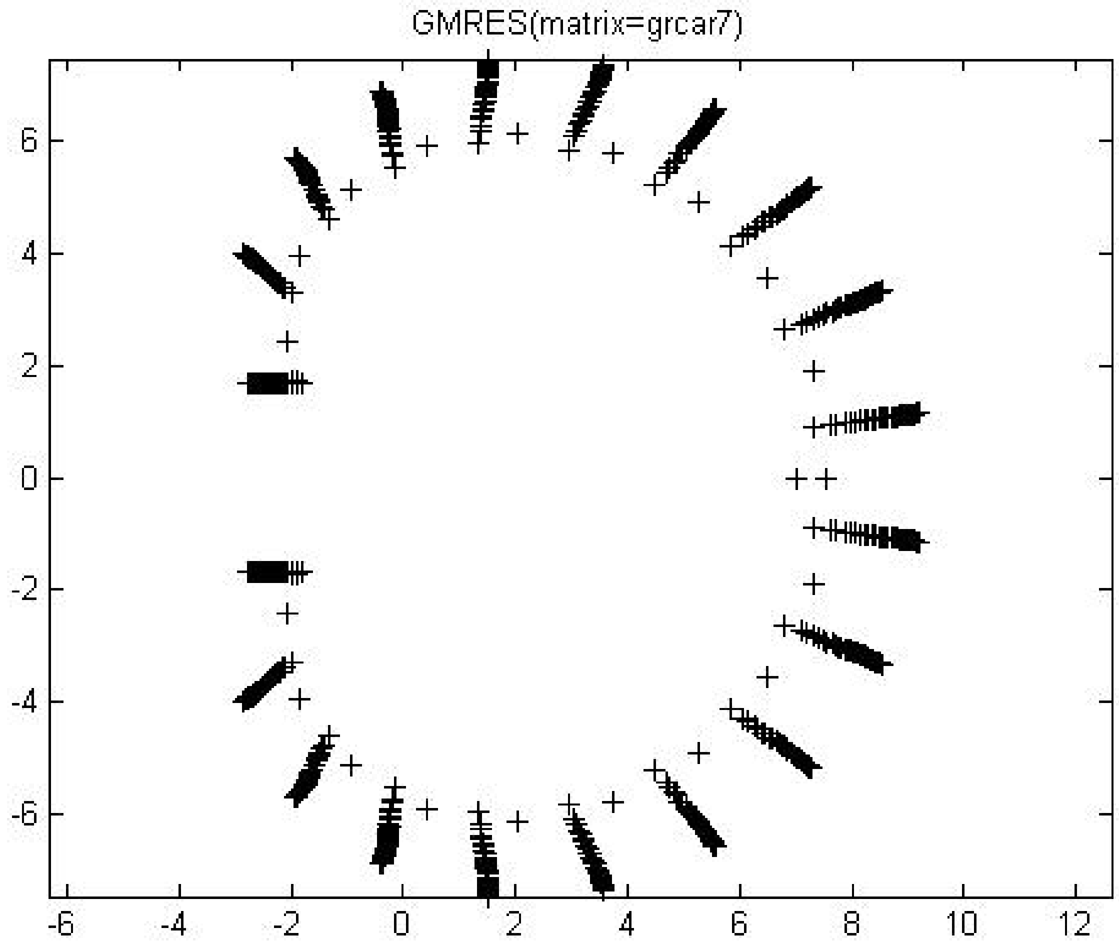
Adding another upper diagonal (making the matrix more non-normal)

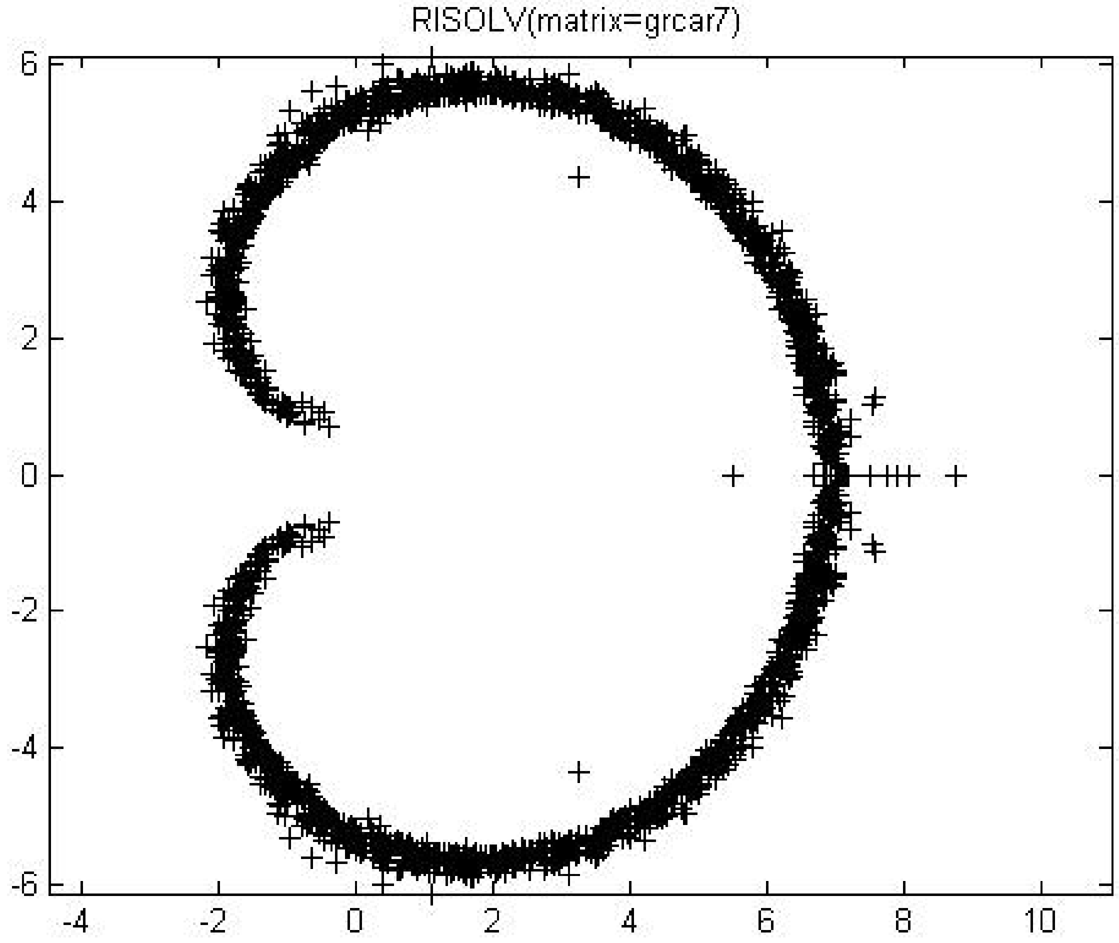
$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & & \\ -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\ & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots\ddots \end{pmatrix}$$

The eigenvalues are



and the $\{z_{ij}\}$ are


GMRES(matrix=grcar7)
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6
8
10
12



Due to the fact that now the pseudospectra of the matrix almost surround the origin, solving the linear system iteratively becomes a much more difficult task. While GMRES stagnated completely, it took 1400 matrix-vector multiplications for RISOLV to reach an accuracy of 10^{-4}.

Example 2: Few linear systems with matrices taken from Matrix-Market

Table 1

matrix	Risolv	Gmres
	mat-vecs	mat-vecs
ADD20	273	1070
MEMPLUS	994	3973
ORSIRR1	919	5186
ORSIRR2	615	2380
ORSREG1	73	79
RAEFSKY1	495	5264
SAYLR4	1028	-

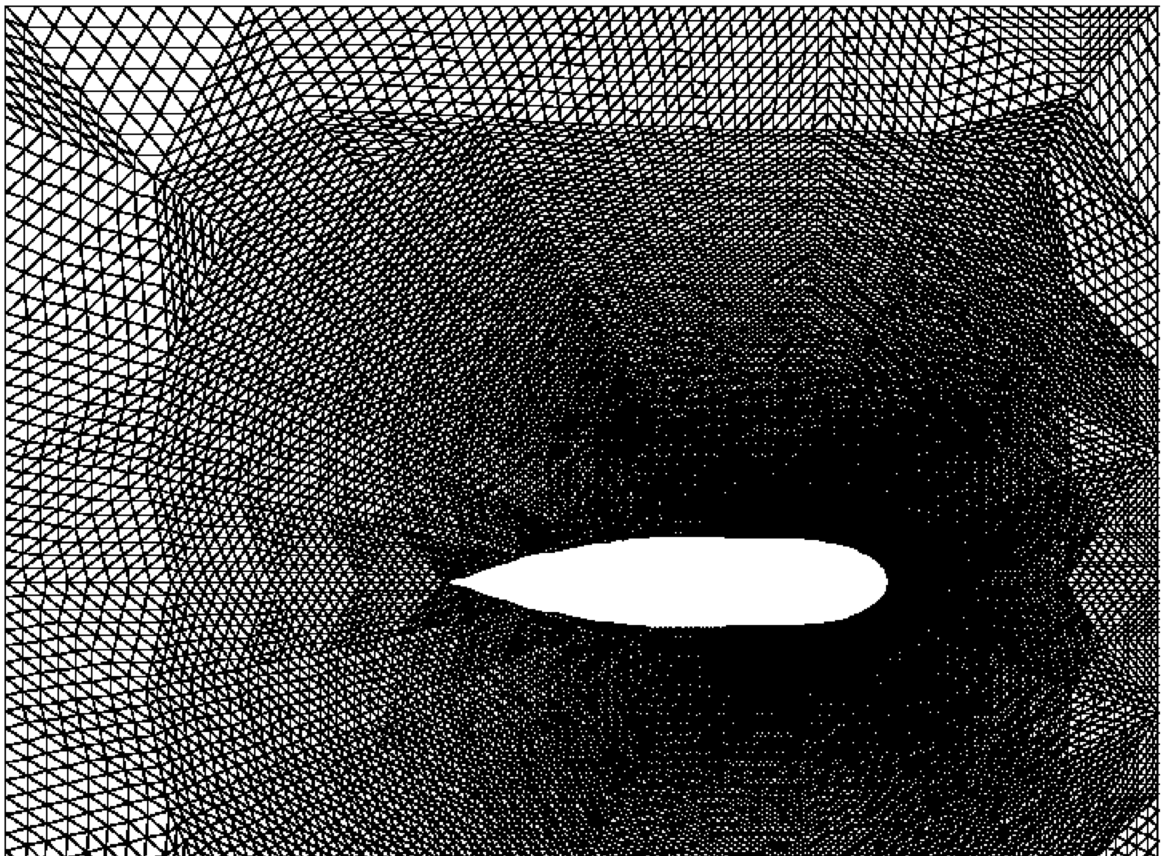
Example 3: Solving $Ax^i = b^i, \qquad 1 \le i \le s.$

In this experiment we have solved 10 linear systems with different r.h.s vectors where A is the ORSIRR1 matrix and b^i are random vectors. Here are the results:

Table 2

no system	mat-vecs	inn-prod
1	905	10923
2	999	2591
3	902	77
4	900	35
5	904	44
6	905	20
7	905	20
8	904	5
9	904	5
10	904	5

Example 3: Convection-Diffusion (taken from SPARSKIT2 of Y. SAAD)



In this case $N = 76800$ and $NNZ = 535553$. We have used ILUT as preconditioner. Drop tolerance of 0.1 was good enough for RISOLV resulting in 624912 size of preconditioner. Drop tolerance of 0.0001 was needed for GMRES resulting in 4752955 size of preconditioner.