On solving mixed sparse-dense linear least-squares problems Part II: A preconditioned conjugate gradient method

Jennifer Scott

University of Reading and STFC Rutherford Appleton Laboratory

Miroslav Tuma

Charles University and Academy of Sciences of the Czech Republic

Sparse Days at CERFACS, 6–8 September 2017

An aside ...

- Although not having attended all *Sparse Days at CERFACS and St.Girons* meetings, I have always appreciated their atmosphere and spirit in general.

An aside ...

- Although not having attended all *Sparse Days at CERFACS and St.Girons* meetings, I have always appreciated their atmosphere and spirit in general.
- I am aware that for all of this I am greatly indebted to Iain!

An aside ...

- Although not having attended all *Sparse Days at CERFACS and St.Girons* meetings, I have always appreciated their atmosphere and spirit in general.
- I am aware that for all of this I am greatly indebted to Iain!
- Having spent half a year at CERFACS in 1998, je suis tambien en peu Toulousain, and always like to return to the South of France.
- Iain, Happy Birthday and many more healthy and productive years!

Introduction

Recall the Linear least squares problem (LS):

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|Ax - b\|_2,$$

where $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ is large and sparse and $b \in \mathbb{R}^m$.

Focus again on A having small number of "dense" rows.

Introduction

Recall the Linear least squares problem (LS):

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|Ax - b\|_2,$$

where $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ is large and sparse and $b \in \mathbb{R}^m$.

Focus again on A having small number of "dense" rows.

Assume the same notation as in the previous talk:

$$A = \begin{pmatrix} A_s \\ A_d \end{pmatrix}$$

where $A_s \in \mathbb{R}^{m_s \times n}$ is sparse but rows of $A_d \in \mathbb{R}^{m_d \times n}$ are dense $(m_s \gg m_d)$. Then

$$C = \begin{pmatrix} A_s^T & A_d^T \end{pmatrix} \begin{pmatrix} A_s \\ A_d \end{pmatrix} = A_s^T A_s + A_d^T A_d$$

Here we consider preconditioned iterative methods

- Preconditioner based on implicitly approximating $C^T C = A^T A$
- Both the sparse part A_s and the dense part A_d should be considered in such approximations (but see the results for unpreconditioned LSQR in Avron, Ng, Toledo, 2009)
- The construction then uses both sparse and dense techniques.
- All of this heavily depends on foundations within the framework of direct methods based on the work of Björck, George, Heath, Ng, Duff and many others.

Here we consider preconditioned iterative methods

- Preconditioner based on implicitly approximating $C^T C = A^T A$
- Both the sparse part A_s and the dense part A_d should be considered in such approximations (but see the results for unpreconditioned LSQR in Avron, Ng, Toledo, 2009)
- The construction then uses both sparse and dense techniques.
- All of this heavily depends on foundations within the framework of direct methods based on the work of Björck, George, Heath, Ng, Duff and many others.
- Crucial practical step: applying factorization $C_s = L_s L_s^T$ to the problem (see, Bjorck, 1986):
- Giving the following equivalent problem

$$\min_z \left\| \begin{pmatrix} B_s \\ B_d \end{pmatrix} z - \begin{pmatrix} b_s \\ b_d \end{pmatrix} \right\|_2, \tag{1}$$

$$B_s = A_s L_s^{-T}, \quad B_d = A_d L_s^{-T}, \quad z = L_s^T x$$

We consider also $C_s \approx L_s L_s^T$.

Preconditioner construction: background 1

- Solving the partial sparse problem $\min_{x_s} \|A_s x_s - b_s\|_2$ approximately
- Completing the result to the exact solution of the original LS problem
- The construction uses residuals that are available in Krylov space accelerators.

Theorem

If $C_s = L_s L_s^T$ and ξ_1 is an approximate solution to the problem $\min_z \|B_s z - b_s\|_2$, the exact least squares solution of the equivalent problem above can be written as $z = \xi_1 + \Gamma_1$, where $\rho_s = b_s - B_s \xi_1$ and $\rho_d = b_d - B_d \xi_1$ and

$$\Gamma_1 = B_s^T \rho_s + B_d^T (I_{m_d} + B_d B_d^T)^{-1} (\rho_d - B_d B_s^T \rho_s). \qquad (2)$$

Preconditioner construction: background 2

- Solving the partial sparse problem $\min_{x_s} \|A_s x_s - b_s\|_2$ exactly
- Completing the result to the exact solution of the original LS problem
- Note that the construction uses residuals that are available in Krylov space accelerators.

Theorem

If $C_s = L_s L_s^T$ and ξ_1 minimizes $\|B_s z - b_s\|_2$ exactly, the exact least squares solution of problem (1) can be written as $z = \xi_1 + \Gamma_1$, , $\rho_s = b_s - B_s \xi_1$ and $\rho_d = b_d - B_d \xi_1$ and

$$\Gamma_1 = B_d^T (I_{m_d} + B_d B_d^T)^{-1} \rho_d. \tag{3}$$

- More possibilities: also the dense part could be approximated
- In fact, ξ_1 can be chosen to solve approximately $\min_\xi \|B_d \xi - b_d\|_2$ instead of $\min_\xi \|B_s \xi - b_s\|_2$

Preconditioned iterations: CGLS1

- Preconditioning reuses some computations of CGLS1.

Algorithm

Preconditioned CGLS1 algorithm $(A_s,\ A_d,\ A_s^T A_s \approx \tilde{L}_s \tilde{L}_s^T;\ z = M^{-1}s))$

0. $r_s^{(0)} = b_s - A_s x^{(0)},\ r_d^{(0)} = b_d - A_d x^{(0)},\ w_s^{(0)} = A_s^T r_s^{(0)},\ w_d^{(0)} = A_d^T r_d^{(0)},$

   $z^{(0)} = M^{-1}(w_s^{(0)} + w_d^{(0)}),\ p^{(0)} = z^{(0)}$

1. **for** $i = 1 : nmax$ **do**

2. $q_s^{(i-1)} = A_s p^{(i-1)},\ q_d^{(i-1)} = A_d p^{(i-1)}, \alpha = \frac{(w_s^{(i-1)} + w_d^{(i-1)}, z^{(i-1)})}{(q_s^{(i-1)}, q_s^{(i-1)}) + (q_d^{(i-1)}, q_d^{(i-1)})}$

3. $x^{(i)} = x^{(i-1)} + \alpha p^{(i-1)},\ r_s^{(i)} = r_s^{(i-1)} - \alpha q_s^{(i-1)},\ r_d^{(i)} = r_d^{(i-1)} - \alpha q_d^{(i-1)}$

4. $z^{(i)} = M^{-1}(A_s^T r_s^{(i)} + A_d^T r_d^{(i)}),\ \beta = \dfrac{(w_s^{(i)} + w_d^{(i)}, z^{(i)})}{(w_s^{(i-1)} + w_d^{(i-1)}, z^{(i-1)})}$

5. $p^{(i)} = z^{(i)} + \beta p^{(i-1)}$

6. **end do**

Experimental evaluation

- Stopping criterion

  C: Stop if $\|r\|_2 < \delta_1$ or $\frac{\|A^T r\|_2}{\|r\|_2} < \frac{\|A^T r_0\|_2}{\|r_0\|_2} * \delta_2$,
  r residual, r_0 initial residual, $\delta_1 = 10^{-8}$ and $\delta_2 = 10^{-6}$.

- Intel(R) Core(TM) i5-4590 CPU at 3.30 GHz, 12 GB of internal memory. Visual Fortran Intel(R) 64 XE compiler

- Most of the matrices from the University of Florida Sparse Matrix Collection
- A prescaled by normalizing its columns
- A row of A considered as dense if the number of entries in the row either exceeds 100 times the average number of entries in a row or is more than 4 times greater than the number of entries in any row in the sparse part A_s.

Experimental evaluation

Table: Statistics: (density= $nnz(C)/n^2$)

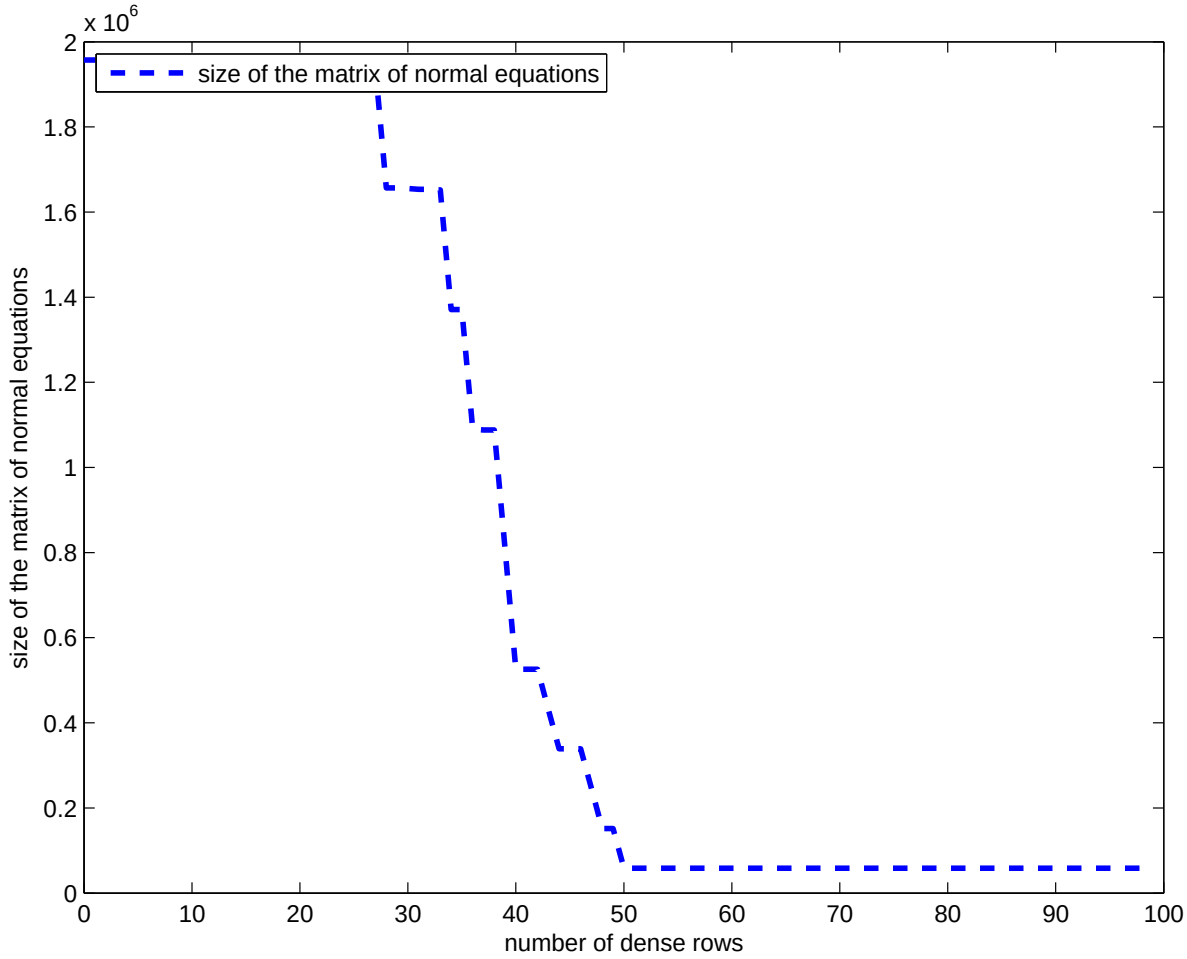
Identifier	m	n	$nnz(A)$	$nnz(C)$	*density*
aircraft	7,517	3,754	20,267	1.4×10^6	0.200
lp_fit2p	13,525	3,000	50,284	4.5×10^6	1.000
scrs8-2r	27,691	14,364	58,439	6.2×10^6	0.143
sctap1-2b	33,858	15,390	99,454	2.6×10^6	0.050
scsd8-2r	60,550	8,650	190,210	2.0×10^6	0.100
scagr7-2r	62,423	35,213	123,239	2.2×10^7	0.036
sc205-2r	62,423	35,213	123,239	6.5×10^6	0.010
sctap1-2r	63,426	28,830	186,366	9.1×10^6	0.050
scfxm1-2r	65,943	37,980	221,388	8.3×10^5	0.014
world	67,147	34,506	198,883	3.1×10^5	0.001
neos1	133,743	131,581	599,590	1.7×10^8	0.027
neos2	134,128	132,568	685,087	2.3×10^8	0.033
stormg2-125	172,431	66,185	433,256	1.0×10^6	0.002
PDE1	270,595	271,792	990,587	1.6×10^{10}	0.670
neos	515,905	479,119	1,526,794	5.3×10^8	0.034
stormg2_1000	1,377,306	528,185	3,459,881	4.2×10^7	0.002
cont1_l	1,921,596	1,918,399	7,031,999	8.2×10^{11}	0.667

Experimental evaluation

	Dense rows not exploited				Dense rows exploited				
Identifier	*size_p*	*T_p*	*Its*	*T_i*	m_d	*size_ps*	*T_p*	*Its*	*T_i*
aircraft	22,509	0.09	44	0.02	17	3,750	0.01	1	0.01
lp_fit2p	17,985	0.26	‡	‡	25	4,940	0.09	1	0.01
scrs8-2r	86,169	0.94	380	0.50	22	36,385	0.01	1	0.02
sctap1-2b	92,325	0.39	639	0.69	34	68,644	0.01	1	0.01
scsd8-2r	51,885	0.25	90	0.11	50	51,855	0.05	7	0.02
scagr7-2r	197,067	3,34	244	0.53	7	152,977	0.06	1	0.01
sc205-2r	211,257	1.56	72	0.19	8	104,022	0.08	1	0.01
sctap1-2r	172,965	1.47	673	1.90	34	127,712	0.03	1	0.01
scfxm1-2r	227,835	0.59	187	0.51	58	227,823	0.14	33	0.23
neos1	789,471	†	†	†	74	789,471	5.27	132	3.71
neos2	†	†	†	†	90	795,323	5.46	157	4.84
stormg2-125	395,595	0.27	‡	‡	121	7,978,135	0.22	16	0.29
PDE1	†	†	†	†	1	1,623,531	12.7	696	1.28
neos	†	†	†	†	20	2,874,699	4.93	232	15.0
stormg2_1000	3,157,095	19.1	‡	‡	121	3,125,987	19.1	18	2.92
cont1_l	†	†	†	†	1	11,510,370	4.82	1	0.33

Moving rows one by one from A_s to A_d

SCSD8-2r_a: size of C_s



Moving rows one by one from A_s to A_d

SCSD8-2r_a: iteration counts + $size\_p/size(A^T A)$

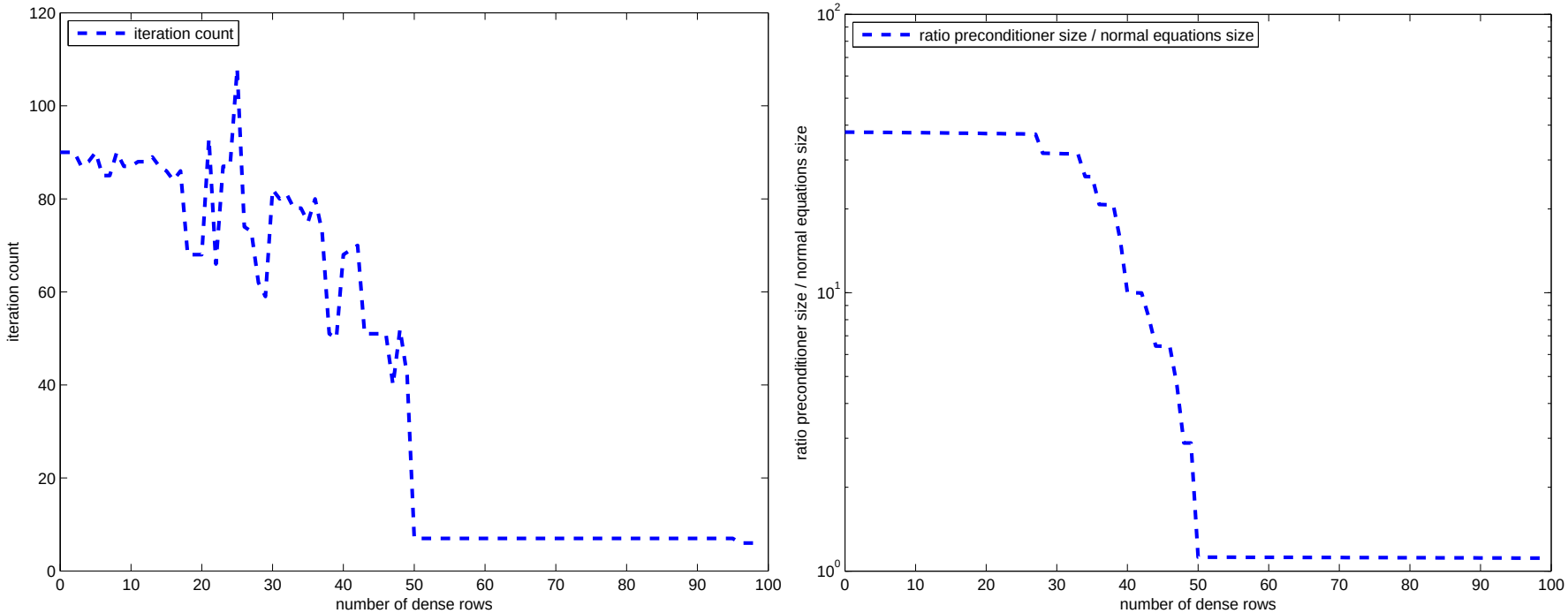


Figure: Problem $Meszaros/scsd8-2r$. Iteration counts (left), and ratio of the preconditioner size to the size of $A^T A$ (right) as the number of dense rows that are removed from A is increased.

Moving rows one by one from A_s to A_d

SCSD8-2r_a: timings

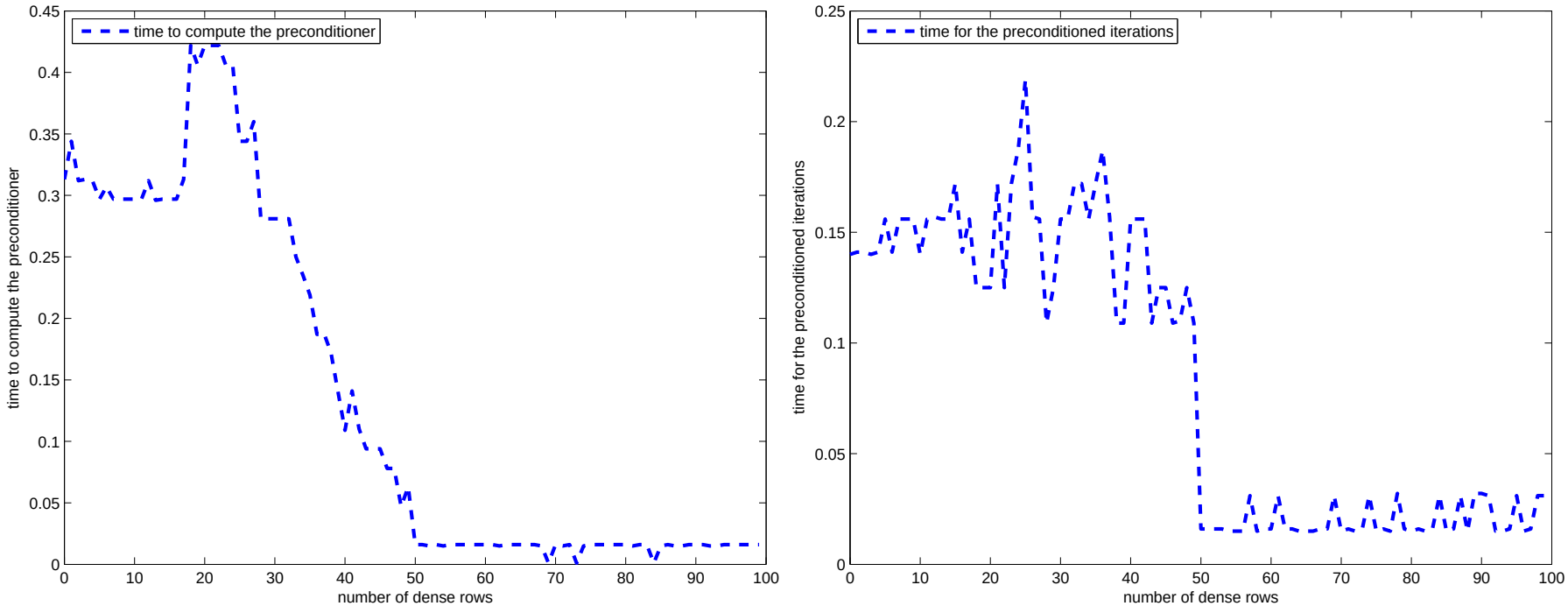


Figure: Problem *Meszaros*/*scsd*8 – 2*r*. Time to compute the preconditioner (left) and time for CGLS (right) as the number of dense rows that are removed from A is increased.

Problem of null columns in A_s after removing A_d

$$A = \begin{pmatrix} A_1 & A_2 \end{pmatrix} \equiv \begin{pmatrix} A_{s_1} & \\ A_{d_1} & A_{d_2} \end{pmatrix}, \tag{4}$$

Problem of null columns in A_s after removing A_d

$$A = \begin{pmatrix} A_1 & A_2 \end{pmatrix} \equiv \begin{pmatrix} A_{s_1} & \\ A_{d_1} & A_{d_2} \end{pmatrix}, \tag{4}$$

- As mentioned in the previous talk: solution can be expressed as a combination of partial solutions of $\min_z \|A_1 z - b\|_2$ and $\min_W \|A_1 W - A_2\|_F$.

Problem of null columns in A_s after removing A_d

$$A = \begin{pmatrix} A_1 & A_2 \end{pmatrix} \equiv \begin{pmatrix} A_{s_1} & \\ A_{d_1} & A_{d_2} \end{pmatrix}, \tag{4}$$

- As mentioned in the previous talk: solution can be expressed as a combination of partial solutions of $\min_z \|A_1 z - b\|_2$ and $\min_W \|A_1 W - A_2\|_F$.
- What about some other preprocessing?
  - E.g., sparsifying the dense rows
  - Such strategy called stretching discussed by Grcar (1990), Vanderbei (1991), Alvarado (1997), Adler (2000), Adler, Björck (2000).

Matrix stretching for the LS problems: generic example

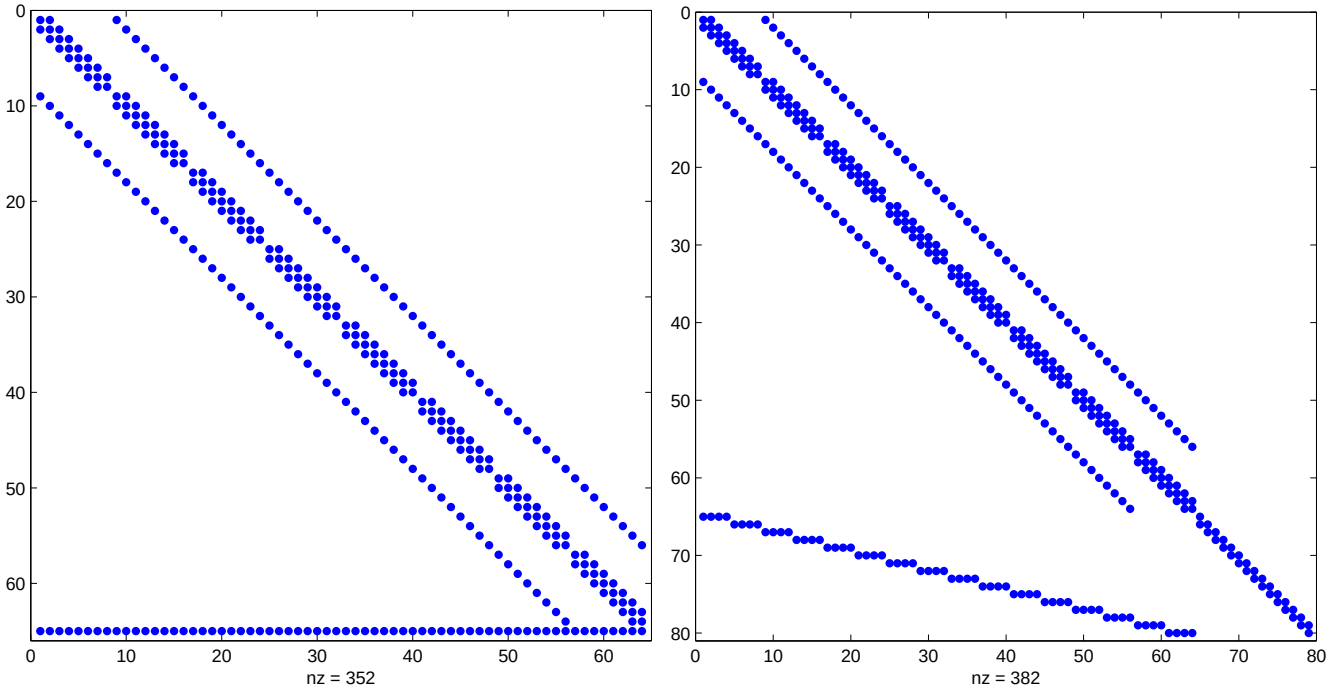


Figure: Matrices based on discrete 2D Laplacian: unstretched (left), stretched (right).

Matrix stretching for the LS problems: generic example

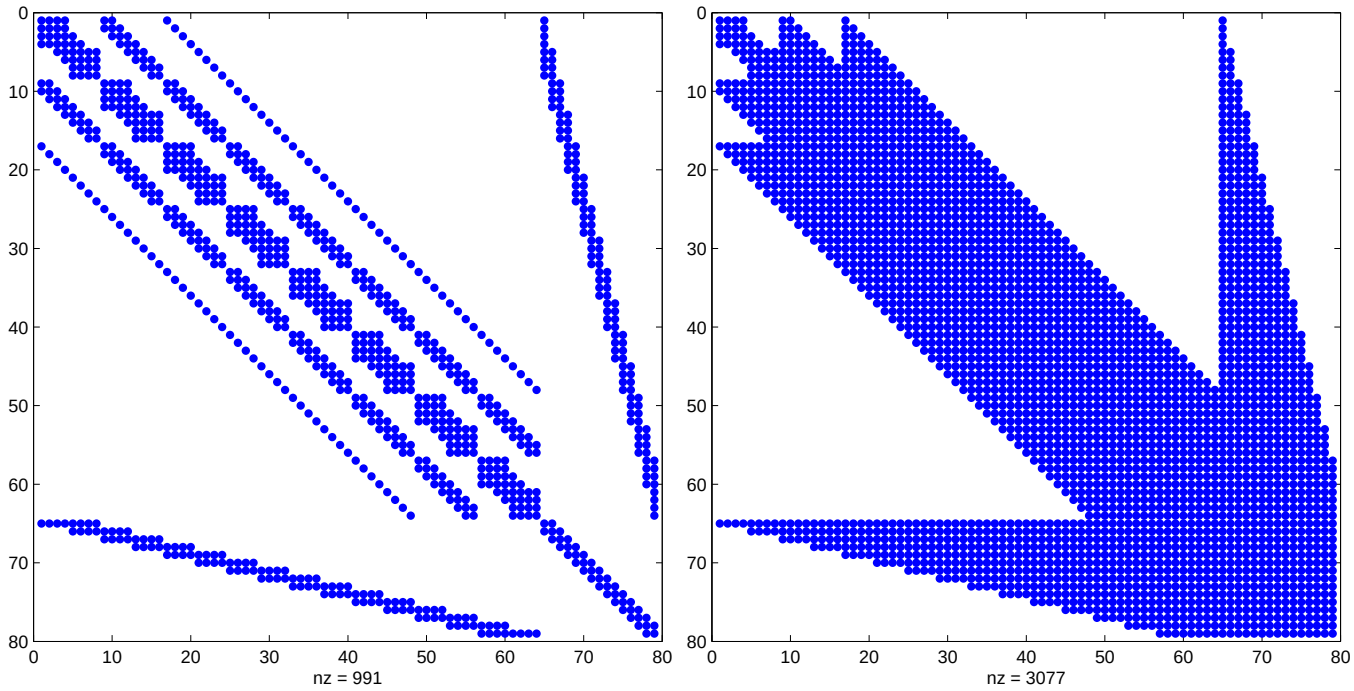


Figure: $A^T A$ (left) and its Cholesky factor (right).

Matrix stretching for the LS problems: generic example

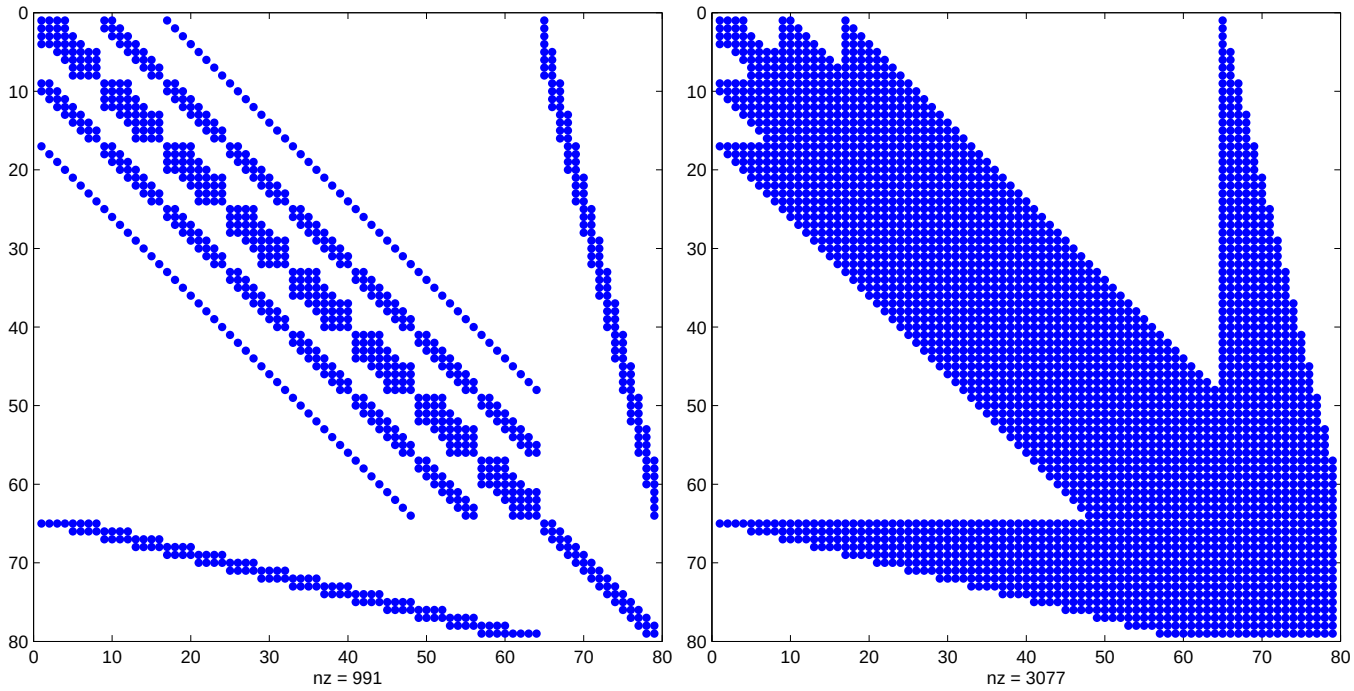


Figure: $A^T A$ (left) and its Cholesky factor (right).

- Bad, but reorderings can help

Matrix stretching for the LS problems: fill-in

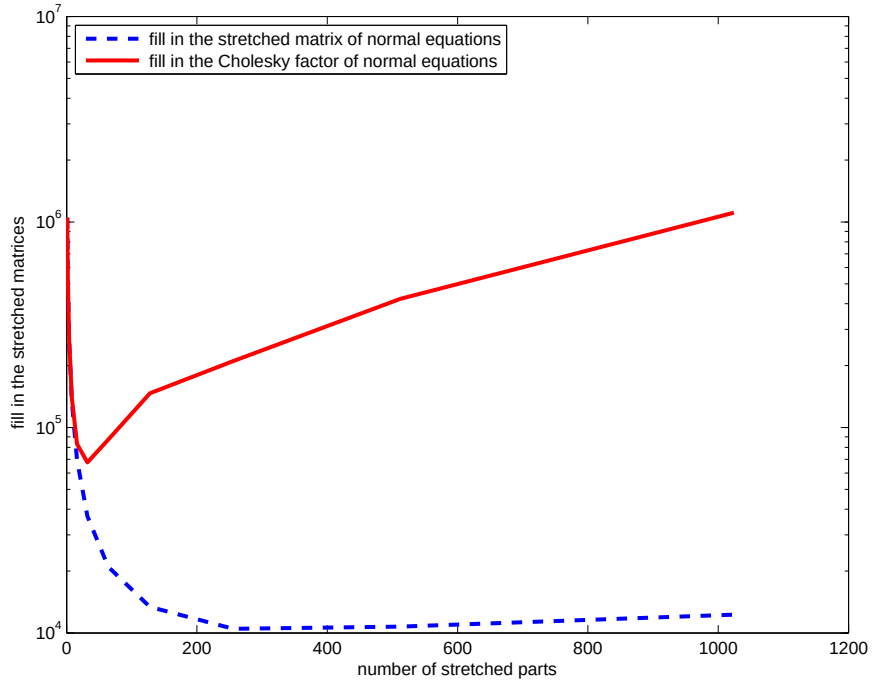


Figure: Fill-in in $A^T A$ and its Cholesky factor (triangular parts).

Matrix stretching for the LS problems: fill-in

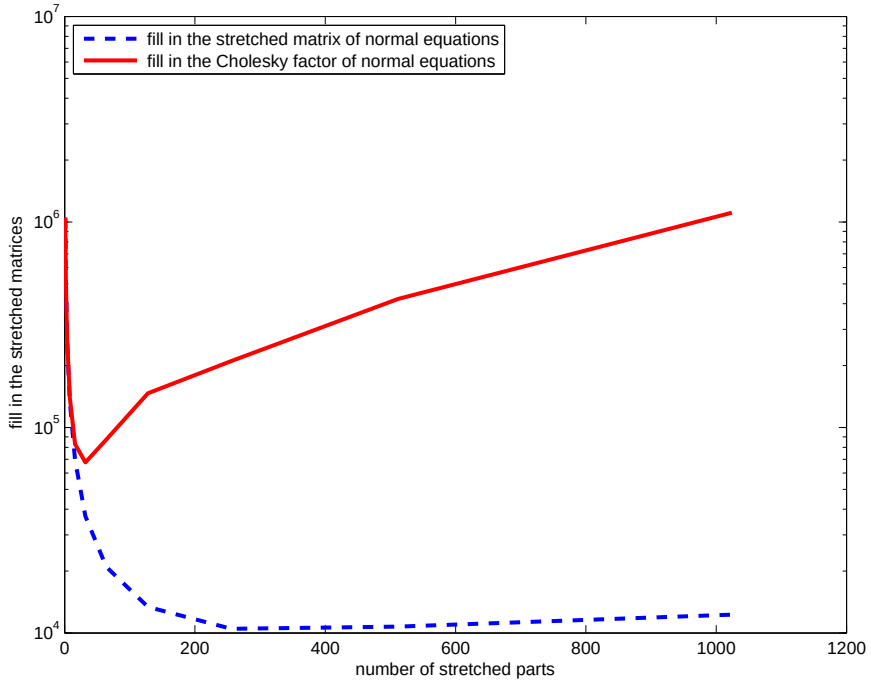


Figure: Fill-in in $A^T A$ and its Cholesky factor (triangular parts).

- Worse, but still not tragic ... Again, reorderings can help.

Matrix stretching for the LS problems: fill-in

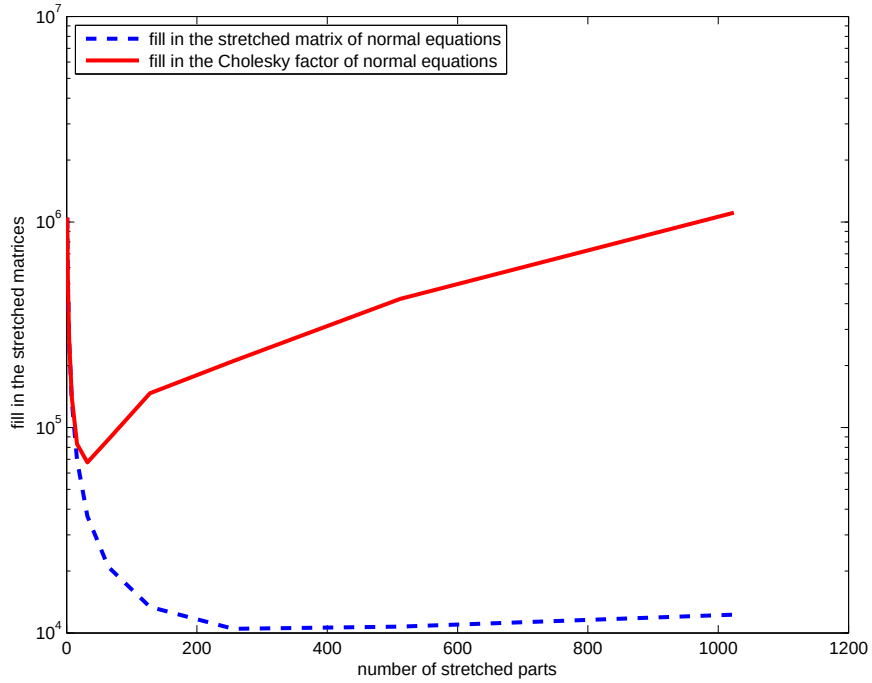


Figure: Fill-in in $A^T A$ and its Cholesky factor (triangular parts).

- Worse, but still not tragic ... Again, reorderings can help.
- More general cases: problems with both fill-in and conditioning (despite the nice result by Adlers and Björck).

Matrix stretching for the LS problems: fill-in

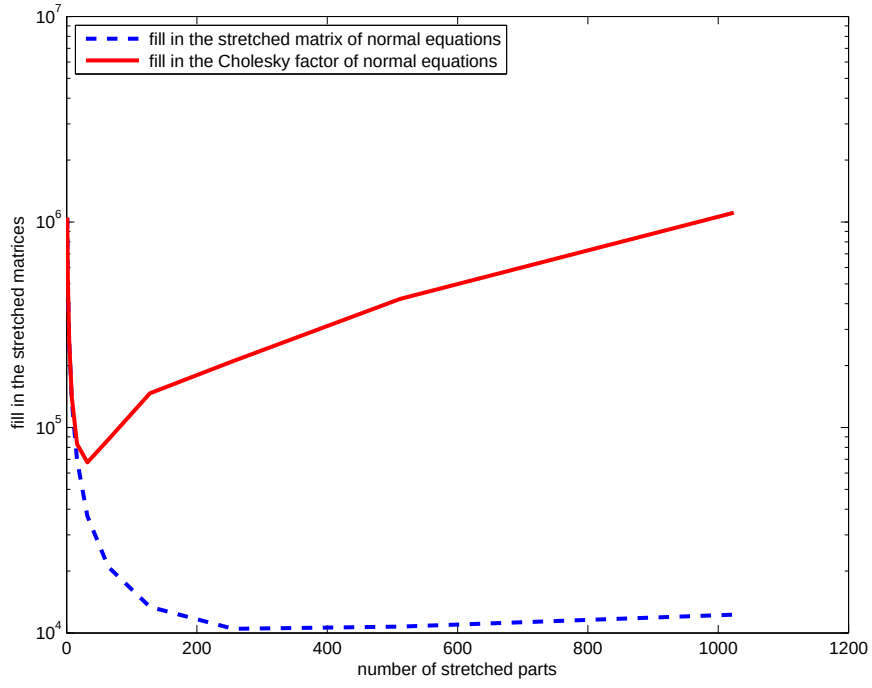


Figure: Fill-in in $A^T A$ and its Cholesky factor (triangular parts).

- Worse, but still not tragic ... Again, reorderings can help.
- More general cases: problems with both fill-in and conditioning (despite the nice result by Adlers and Björck).
- The problem with fill-in is primarily not in additional memory, but in contributing to loss of information in preconditioners!

Our solution (sketched here): make a bridge between A_s and A_d

- Also positive results: stretching just to make a bridge between A_s and A_d
- Matrix AIRCRAFT, stretching used to replace the combination of the partial solutions.
- The decrease in the left automatic due to the IC.

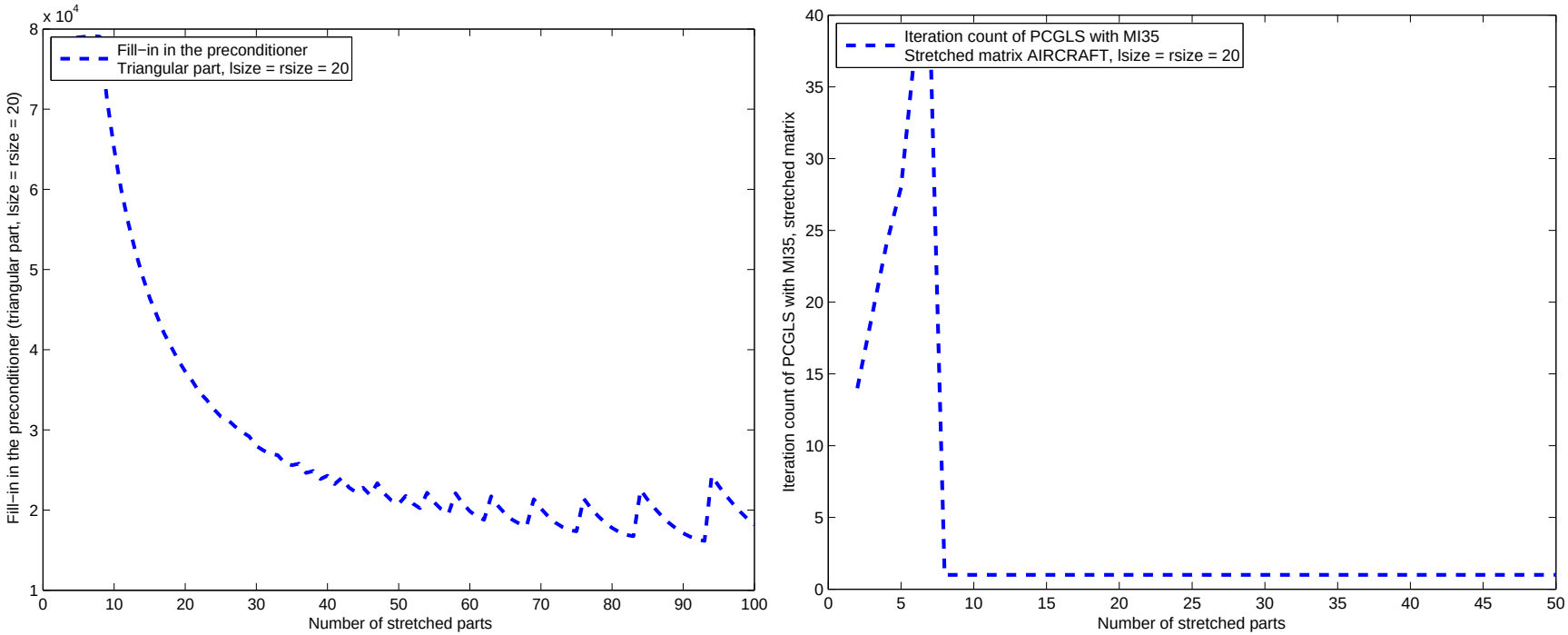


Figure: Stretching only for the bridge. Fill in $A^T A$ (left), iteration count (right)

Conclusions

- Solving linear least squares problems where A has a number of dense rows.

Conclusions

- Solving linear least squares problems where A has a number of dense rows.
- A new approach that processes the dense rows separately within a conjugate gradient method

Conclusions

- Solving linear least squares problems where A has a number of dense rows.
- A new approach that processes the dense rows separately within a conjugate gradient method
- Not all the formulas above work in practice !!!! In our case, the best has been the simplest one.

Conclusions

- Solving linear least squares problems where A has a number of dense rows.
- A new approach that processes the dense rows separately within a conjugate gradient method
- Not all the formulas above work in practice !!!! In our case, the best has been the simplest one.
  - Dense rows must be treated separately
  - Dense rows must be considered
  - Problem of zero columns in A_s solved by limited use of the stretching.

Last but not least

Thank you for your attention!