Unsymmetric Orderings Using A Constrained Markowitz Scheme

Patrick R. Amestoy [*], Xiaoye S. Li [†], Stéphane Pralet [‡]

Introduction

We consider LU factorization of a sparse unsymmetric matrix A based on the three-phase approaches (analysis, factorization, solve). The analysis phase transforms A into $\bar{A}$ with better properties for sparse factorization. It exploits the structural information to reduce the fill-ins in the LU factors and exploits the numerical information to reduce the amount of numerical pivoting needed during factorization. Two consecutive treatments are commonly used for these two objectives. Firstly, scaling and maximum transversal algorithms are used to transform A into A_1 with large entries on the diagonal. Secondly, a symmetric fill-reducing ordering, which preserves the large diagonal, is used to permute A_1 into A_2 so that the factors of A_2 are sparser than those of A_1. Note that during factorization, numerical instabilities can still occur and will be handled either by partial pivoting resulting in extra fill-in in the factor matrices or by static pivoting resulting in a potentially less accurate factorization.

During analysis, the numerical treatment requires the fill-reducing ordering to be symmetric, and the structural phase does not have numerical information to correct any wrong numerical decisions. To avoid these two drawbacks, we present an approach which mixes the numerical and structural phases. Based on a numerical pre-treatment of the matrix we build at each step k of the elimination a set of numerically acceptable pivots, referred to as matrix C^k that may contain off-diagonal entries. We then compute an unsymmetric ordering taking into account both the structure of A and the numerical information in C^k.

Main components of our unsymmetric ordering

Let A^1 be the original matrix of order n and A^k be the reduced matrix after eliminating the first $k-1$ pivots (not necessarily on the diagonal). Let C^1 be such that $\mathcal{P}attern(C^1) \subset \mathcal{P}attern(A^1)$. At each step k of the algorithm we select the best pivot for a given metric such that $p \in \mathcal{P}attern(C^k)$. Matrix A^k is updated (remove row and column of the pivot and add fill-ins in the Schur complement). Matrix C^k is updated so that C^{k+1} is structurally nonsingular and $\mathcal{P}attern(C^{k+1})$ is included in the reduced matrix C^k after eliminating pivot p. This implies that $\mathcal{P}attern(C^{k+1}) \subset \mathcal{P}attern(A^{k+1})$.

During the unsymmetric ordering, the k^{th} pivot $p \in C^k$ can be selected using various structural local heuristics (approximation of Markowitz count, approximate minimum fill, etc.). To update matrix C we have implemented three strategies. The first 2 strategies require only the pattern of C^k whereas the third strategy also requires handling the numerical values.

- Strategy 1 (`strat1`): add entries in C^k only to maintain the structural non-singularity.
- Strategy 2 (`strat2`): perform all structural updates in C^k.
- Strategy 3 (`strat3`): perform all numerical update on C^k. (A pivot can be selected if and only if $|c_{ij}^{(k+1)}| > th$.)

In our experiment, we build C^1 as follows. First, `MC64` (Duff and Koster 1999) is applied on A to obtain the diagonal scaling factors D_r and D_c (with the property $|d_{r_i} a_{ij} d_{c_j}| \leq 1$) and the maximum weighted matching $\mathcal{M}$. We then set $C^1 = (c_{ij})$ such that if $|d_{r_i} a_{ij} d_{c_j}| > threshold$, $c_{ij} = d_{r_i} a_{ij} d_{c_j}$, otherwise $c_{ij} = 0$. Further dropping in C^1 can be applied to limit the memory and the complexity of the ordering phase.

[*] Patrick.Amestoy@enseeiht.fr, ENSEEIHT-IRIT, 2, rue Camichel, BP 7122 - F 31071 Toulouse Cedex 7, France.
[†] xsli@lbl.gov, Lawrence Berkeley National Lab, MS 50F-1650, 1 Cyclotron Rd., Berkeley, CA 94720.
[‡] Stephane.Pralet@cerfacs.fr, CERFACS, 42, av. G. Coriolis, 31057 Toulouse Cedex 01, France.

Table 1: Comparison of MC64+DMLS with a MATLAB implementation of our unsymmetric ordering (Approximate minimum fill metric is used in both cases).

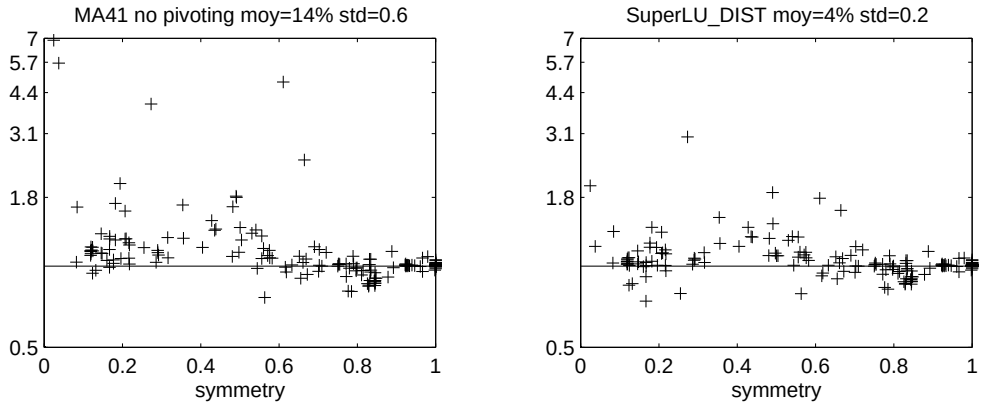
Matrix	strat2				strat3				MC64+DMLS			
	lu th=0		*lu th*=0.01		*lu th*=0		*lu th*=0.01		*lu th*=0		*lu th*=0.01	
	nnz	*err*	nnz	*err*	nnz	*err*	nnz	*err*	nnz	*err*	nnz	*err*
MAHINDAS	12574	7e-16	12574	7e-16	12574	7e-16	12574	7e-16	15699	4e-16	15699	4e-16
LHR01	51829	1e-10	52613	1e-14	52216	5e-15	52216	5e-15	59070	2e-14	59023	6e-15
B_DYN	8055	7e-15	8074	1e-15	8071	7e-16	8071	7e-16	8795	4e-16	8795	4e-16
B2_SS	7392	NaN	7478	7e-14	7484	4e-15	7484	4e-15	8393	NaN	8455	3e-15
RADFR1	31273	NaN	61681	7e-10	34986	3e-11	34030	1e-11	31373	NaN	32935	5e-11
GRE_1107	42864	1e-10	46495	3e-11	42887	5e-11	44662	7e-12	70113	4e-11	70721	2e-11
EX23	94837	1e14	109531	6e-14	93744	8e-14	93744	8e-14	131583	2e01	134791	3e-14

In Table 1, we compare the number of non-zeros in the factors (nnz) and the forward error ($err = ||xtrue - x||_1/||x||_1$) of strategies 2 and 3 with an approach combining MC64 and DMLS (Diagonal Markovitz with Local Symmetrization) ordering (Amestoy, Li and NG 2003). We use MATLAB *LU* factorization both with no pivoting (th=0) and with a more common pivoting threshold value $th = 0.01$ (to evaluate the extra fill-ins due to numerical pivoting). We see that, except for RADFR1, with strategies 2 and 3 the extra freedom given to select off-diagonal pivots in C^k results in a reduction of the fill-in with respect to MC64+DMLS approach. The increase in the fill-ins while increasing the value of th is quite reasonable, except for RADFR1 with strategy 2, and shows that this extra freedom has not introduced extra numerical problems. Finally, we see that strategy 3 is very stable even when no numerical pivoting is performed during factorization.

Constrained Markowitz with Local Symmetrization

To have an efficient in-place implementation we use a quotient graph representation for matrix A^k, to perform local symmetrization (generalizing the strategy used in DMLS) and to use a pivot selection criterion based on a minimum fill metric. In the first implementation of the algorithm, referred to as CMLS, at each step, we strictly limit the size of C^k. $|C^1|$ is forced to be lower than $4n$. If the memory needed to store C^{k+1} is smaller than the memory needed to store C^1, we perform all the structural updates in C^k (strat2). Otherwise, an in-place implementation of strat1 is used.

Figure 1: Ratio DMLS/CMLS of the factor size. (MC64 is included in DMLS preprocessing).



All the matrices from Tim Davis' collection of order between 1000 and 20000 have been used to run the unsymmetrized multifrontal version of MA41 (Amestoy and Puglisi 2002) without pivoting and the SuperLU_DIST code with static pivoting (Li and Demmel 2003). We see in Figure 1 that using CMLS ordering leads to a significant reduction in the fill-ins compared with the MC64+DMLS algorithm (for matrices with a symmetry smaller than 0.5, the average gain with MA41 is 46% and with SuperLU_DIST is 16%).