# Ordering for Saddle Point Matrices in SparseCholesky

Zero diagonals, dense rows, and the flags of `Inverse`

Joachim Schöberl

NGSolve Technical Note 2026-05 • November 2026

## 1 The problem

`SparseCholesky`, the default direct solver of NGSolve (`inverse="sparsecholesky"`), computes a supernodal factorization $K = LDL^T$ with a diagonal $D$. There is no pivoting: the elimination order is fixed in advance by a minimum degree ordering [1] (class `MinimumDegreeOrdering` in `linalg/order.cpp`), which sees only the graph of the matrix. For symmetric positive definite matrices every order is stable. For saddle point matrices [2]

$$K = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}$$

two things went wrong.

- **Zero pivots.** A dof with zero diagonal that is eliminated before all of its neighbours has the pivot 0. Minimum degree takes dofs with few neighbours first, for example the `L2` dofs of a mixed method. The factorization then contains NaN; no exception is raised.
- **Dense rows.** A `NumberSpace` dof, used for a mean value or another global constraint, couples to all dofs of a field. One vertex of degree $O(n)$ makes every elimination step of the minimum degree algorithm cost $O(n)$, in total $O(n^2)$. For an $H^1$ problem with 186 000 dofs and one such row the ordering took three minutes (Table 1).

Both are solved inside the ordering, by constraints on the elimination order. The factorization kernels are unchanged, and a matrix without zero diagonal entries and without dense rows gets exactly the ordering it got before.

## 2 Zero diagonal: the wait rule

Every dof gets a level. A dof with nonzero diagonal entry has level 0. A dof with zero diagonal has level 1 + the smallest level of its neighbours, which is its distance in the graph to the nearest level-0 dof (breadth-first search). For the Stokes problem with a mean value constraint the velocity has level 0, the pressure level 1, and the multiplier of the mean value, which couples only to the pressure, level 2.

**Rule:** a dof enters the priority queue of the minimum degree algorithm only after all its neighbours of lower level have been eliminated. Until then it keeps a wait counter (`nwait`); eliminating a vertex decrements the counters of its higher-level neighbours (`Release`). Among the dofs that are not waiting, the choice is by minimum degree as before. This is the constrained ordering of Bridson [3]; see also [4].

**Why it is sufficient.** Let $A$ be symmetric positive definite on the level-0 dofs $U$, and let $B$ have full row rank on the level-1 dofs $P$. In an order that follows the rule, a leading principal submatrix with dofs $U_1 \subset U$ and $P_1 \subset P$ is

$$K_1 = \begin{pmatrix} A_{11} & B_1^T \\ B_1 & 0 \end{pmatrix}, \qquad B_1 = B|_{P_1 \times U_1}.$$

All neighbours of $P_1$ are in $U_1$, so $B_1$ consists of complete rows of $B$ and has full row rank; $A_{11}$ is positive definite. Thus every $K_1$ is regular, and $LDL^T$ exists without pivoting. The pivots of $U$ are positive, those of $P$ negative. A singular $A$ whose kernel is fixed by the constraint is not covered by this argument; this case is treated by the placement rules of Section 3.

**Supernodes.** The algorithm merges indistinguishable vertices into supernodes, which are eliminated as one dense block. A waiting vertex is not merged. After the ordering, the vertices inside each supernode are sorted by level.

**Variants.** The flag `late` selects which dofs wait.

- `late="zero"` (default): diagonal entry exactly zero.

- `late="nonpos"`: in addition, dofs with negative diagonal get the fixed level 1 (real matrices only). This is for matrices $\left(\begin{smallmatrix} A & B^T \\ B & -C \end{smallmatrix}\right)$ and for matrices after static condensation, where a zero diagonal is polluted by roundoff. It is not the default, because matrices with diagonal entries of both signs and no saddle point structure, such as $K - \omega^2 M$, would get constraints that only cost fill.
- `late="off"`: no constraints.
- `waitfrac` $= \theta < 1$: a dof waits only for the fraction $\theta$ of its lower-level neighbours. This reduces the fill, but the argument above is lost; it is a heuristic (Table 2).

# 3 Dense rows

**Detection.** A vertex is dense if its degree exceeds $\max(16,\ \texttt{densefac} \cdot \sqrt{n})$, with `densefac` $= 10$ as in AMD [5]. If more than 256 vertices qualify, none is treated as dense. The flag `lastdofs` adds dofs by hand. Dense vertices are removed from the graph before the minimum degree algorithm runs.

**Structure.** The nonzero pattern of the factor is afterwards completed exactly: a dense dof $d$ belongs to the structure of a supernode $S$ if and only if $d$ is adjacent to $S$ or to a descendant of $S$ in the elimination tree. One bit per dense dof is propagated up the tree (`AppendDense`). The factor has the entries of the ordering of the reduced graph plus the rows of the dense dofs, nothing more.

**Placement.** The obvious choice, dense dofs last, is wrong. For the Neumann problem with a mean value constraint, $A$ is singular with the constants as kernel. If all $u$-dofs come first, the last of them has the pivot 0. The constraint has to be eliminated while $A$ restricted to the dofs eliminated so far is still regular. Therefore the final supernode is

$$[\ \text{dense dofs},\ \text{root supernode of the elimination tree}\ ].$$

The whole root supernode is taken: holding back only as many dofs as there are constraints was not enough for the six rigid body motions in 3D. If the graph is disconnected (several floating subdomains, each with its own constraint), there are several roots. Then from every root that couples to $m$ dense dofs, the last $2m$ vertices of each level are moved into the final supernode. If this would move more than $\max(64, 4m_{\text{total}})$ vertices, the dense dofs are appended at the end without this care.

**Multipliers that are not dense.** The multiplier of a small floating subdomain has a degree below the threshold. It stays in the graph, has zero diagonal, and by the wait rule would come after its whole subdomain, which gives the same zero pivot. Hence in a root supernode the waiting vertices that have a lower-level neighbour outside of this supernode are moved to its front.

**A bug found on the way.** A `NumberSpace` dof has no stored diagonal entry. The diagonal of the factor was not initialized in this case; `FactorNew` now sets it to zero.

# 4 Flags and a complete example

The options are passed as `flags` to `Inverse`; they are documented in its docstring.

| flag | default | meaning |
|---|---|---|
| `late` | `"zero"` | which dofs are eliminated after their neighbours: `"zero"`, `"nonpos"`, `"off"` |
| `densefac` | 10 | rows with more than `densefac`$\cdot\sqrt{n}$ entries are dense; 0 switches the detection off |
| `lastdofs` | `[]` | list of dof numbers treated like dense rows |
| `waitfrac` | 1 | wait only for this fraction of the lower-level neighbours |

Stokes flow with Dirichlet conditions on the whole boundary; the mean value of the pressure is fixed by a `NumberSpace` multiplier. The matrix has zero diagonal for the pressure and the multiplier, and one dense row.

```
from ngsolve import *
from netgen.occ import unit_square
mesh = Mesh(unit_square.GenerateMesh(maxh=0.02))

V = VectorH1(mesh, order=2, dirichlet=".*")
Q = H1(mesh, order=1)
X = V * Q * NumberSpace(mesh)          # velocity, pressure, mean value multiplier
(u, p, lam), (v, q, mu) = X.TnT()

a = BilinearForm(X, symmetric=True)
```

```
a += (InnerProduct(Grad(u), Grad(v)) + div(u)*q + div(v)*p + lam*q + mu*p)*dx
f = LinearForm(X)
f += CF((0, x-0.5))*v*dx

with TaskManager():
    a.Assemble()
    f.Assemble()
    gf = GridFunction(X)
    for flags in [{}, {"late": "off"}, {"densefac": 0}, {"waitfrac": 0.25}]:
        inv = a.mat.Inverse(X.FreeDofs(), inverse="sparsecholesky", flags=flags)
        gf.vec.data = inv * f.vec
        res = (Projector(X.FreeDofs(), True) * (f.vec - a.mat * gf.vec)).Evaluate()
        print(flags, "nze =", inv.nze, " residual =", Norm(res)/Norm(f.vec))
```

The space has 25 777 free dofs. The script prints

```
{} nze = 3020647  residual = 4.97e-14
{'late': 'off'} nze = 2317800  residual = nan
{'densefac': 0} nze = 3020647  residual = 5.02e-14
{'waitfrac': 0.25} nze = 2310346  residual = 4.91e-14
```

(residuals shortened to three digits). `inv.nze` is the number of entries of $L$. Without the dense row detection the result is the same here, since the multiplier has level 2 and is moved by the rule for non-dense multipliers.

# 5 Measurements

All runs on an Apple M4 Pro with `TaskManager`, NGSolve at commit `89946f48b`; unstructured triangular and tetrahedral meshes of the unit square and cube; random right-hand side; relative residual on the free dofs.

| free dofs | default | `densefac=0` | nze(default) / nze(`densefac=0`) |
|---|---|---|---|
| 11 790 | 0.02 s | 0.34 s | 1.04 |
| 46 726 | 0.07 s | 5.68 s | 1.04 |
| 185 906 | 0.31 s | 183.8 s | 1.01 |

Table 1: Time of the call `Inverse` (ordering and factorization) for the Neumann problem with mean value constraint, `H1(order=2)*NumberSpace` on the unit square. The residuals are below $3 \cdot 10^{-14}$ in all runs. With six rigid body constraints on the unit cube (23 625 dofs): 0.20 s and 2.09 s, nze ratio 1.08.

| | | `late="off"` | | default | | `waitfrac=0.25` | |
|---|---|---|---|---|---|---|---|
| problem | free dofs | nze | residual | fill | residual | fill | residual |
| Stokes 2D, $P^2$-$P^1$ | 26 374 | $2.35 \cdot 10^6$ | $9 \cdot 10^{-13}$ | 1.35 | $8 \cdot 10^{-13}$ | 1.01 | $9 \cdot 10^{-13}$ |
| Stokes 3D, $P^2$-$P^1$ | 23 284 | $13.5 \cdot 10^6$ | $1 \cdot 10^{-13}$ | 1.20 | $1 \cdot 10^{-13}$ | 1.00 | $1 \cdot 10^{-13}$ |
| Stokes 2D, $P^4$-$P^3$ | 30 199 | $2.78 \cdot 10^6$ | $5 \cdot 10^{-12}$ | 2.59 | $6 \cdot 10^{-12}$ | 1.03 | $5 \cdot 10^{-12}$ |
| Stokes 2D, $P^2$-$P^1$, mean value | 25 777 | $2.32 \cdot 10^6$ | NaN | 1.30 | $1 \cdot 10^{-12}$ | 1.00 | $2 \cdot 10^{-12}$ |
| mixed Poisson 2D, $RT_1$-$P^1$ | 46 552 | $0.75 \cdot 10^6$ | NaN | 5.77 | $4 \cdot 10^{-13}$ | 0.98 | NaN |
| mixed Poisson 3D, $RT_1$-$P^1$ | 64 320 | $4.90 \cdot 10^6$ | NaN | 6.47 | $2 \cdot 10^{-14}$ | 1.02 | $2 \cdot 10^{16}$ |
| condensed TDNNS 2D, order 3 | 7 296 | $0.54 \cdot 10^6$ | NaN | 1.00 | NaN | `late="nonpos"`: 2.53, | $1 \cdot 10^{-14}$ |

Table 2: Entries of $L$ without constraints, and fill relative to it with the wait rule. Stokes: Dirichlet data on one side, or on the whole boundary with mean value multiplier. Mixed Poisson: `HDiv(order=1, RT=True)*L2(order=1)`, not hybridized. TDNNS: `HDivDiv(3)*HCurl(3)` with `condense=True`.

- **Without constraints** the factorization fails where zero-diagonal dofs have few neighbours: the `L2` dofs of the mixed method, and the pressure next to the multiplier. For continuous pressures it succeeded in these runs; nothing guarantees this.
- **The price is fill.** It is moderate for Taylor–Hood elements and grows with the order. For the mixed method it is a factor 6; there hybridization, which leads to a positive definite matrix, is the better way.
- `waitfrac` removes the extra fill and worked for all Stokes runs, but fails for the mixed method.
- **Condensed matrices.** In the condensed TDNNS matrix 880 of the 7 296 diagonal entries are below $10^{-12}$ in modulus, but none is exactly zero. The default `late="zero"` does not see them, and the factorization fails as before. `late="nonpos"` repairs it.

# 6 Rejected alternatives

- *Dense dofs at the very end, or only $m$ dofs held back for $m$ constraints:* zero pivot for kernel-fixing constraints (Section 3).
- *Sorting by level inside the supernodes only, without the wait rule:* did not repair the failing cases in the experiments of November 2026.
- *`late="nonpos"` as default:* constraints without need for indefinite matrices that are not saddle point matrices.

# 7 Open points

- A zero pivot is not detected. A failed factorization returns NaN without an exception.
- `late="zero"` tests for exact zeros. A tolerance, or a test on the element matrices before condensation, is missing.
- A release rule based on a weighted matching, as in the pivoting strategies of [6, 7], should need less fill than the wait rule, in particular for mixed methods. Not implemented.
- No pivoting inside the final supernode.
- Many floating subdomains with dense multipliers: beyond the limit of Section 3 the dense dofs are appended without the kernel-fixing placement.
- `late="nonpos"` has no effect for complex and for block matrices.

**Code.** Commit `89946f48b` (November 10, 2026): `linalg/order.hpp`, `linalg/order.cpp` (`MinimumDegreeOrdering`, `AppendDense`), `linalg/sparsecholesky.cpp` (flags), `tests/pytest/test_sparsecholesky_saddle.py`.

# References


[1] A. George and J. W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Rev.* 31(1):1–19, 1989.

[2] M. Benzi, G. H. Golub and J. Liesen. Numerical solution of saddle point problems. *Acta Numer.* 14:1–137, 2005.

[3] R. Bridson. An ordering method for the direct solution of saddle-point matrices. Preprint, 2007.

[4] M. Tůma. A note on the $LDL^T$ decomposition of matrices from saddle-point problems. *SIAM J. Matrix Anal. Appl.* 23(4):903–915, 2002.

[5] P. R. Amestoy, T. A. Davis and I. S. Duff. Algorithm 837: AMD, an approximate minimum degree ordering algorithm. *ACM Trans. Math. Software* 30(3):381–388, 2004.

[6] I. S. Duff and S. Pralet. Strategies for scaling and pivoting for sparse symmetric indefinite problems. *SIAM J. Matrix Anal. Appl.* 27(2):313–340, 2005.

[7] O. Schenk and K. Gärtner. On fast factorization pivoting methods for sparse symmetric indefinite systems. *Electron. Trans. Numer. Anal.* 23:158–179, 2006.