Efficient multigrid and data structures for edge-oriented FEM stabilization

Abderrahim Ouazzi and Stefan Turek

Institute of Applied Mathematics, University of Dortmund, 44227 Dortmund, Germany, ouazzi@math.uni-dortmund.de, ture@featflow.de

Summary. We study edge-oriented FEM stabilizations w.r.t. linear multigrid solvers and data structures with the goal to examine the efficiency of such stabilizations due to the extending matrix stencil which is not supported by standard FEM data structures. A new edge-oriented data structure has been developed to support the additional coupling. So, the local element-wise and edge-wise matrices are easily deduced from the global ones. Accordingly, efficient Vanka smoothers are introduced, namely a full cell-oriented and an edge-oriented Vanka smoother so that it becomes possible to privilege edge-oriented stabilization for CFD simulations.

1 Introduction

1.1 Problem Formulation

As a model problem we consider incompressible flow described by the Navier-Stokes equations:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \triangle \boldsymbol{u} + \nabla p = \boldsymbol{f}, \quad \operatorname{div} \boldsymbol{u} = 0 \tag{1}$$

where p is the pressure and $\boldsymbol{u}$ being the velocity. Let us consider the nonstationary (or stationary, without the term $\frac{\partial \boldsymbol{u}}{\partial t}$) Navier-Stokes problem (1) in a bounded domain $\Omega \subset \mathbb{R}^2$, first discretized in time by a standard numerical solution method for ODEs. The θ-scheme, as for instance backward Euler or Crank-Nicholson or the Fractional-step-θ-scheme, yields a sequence of boundary value problems of the following form [2]:
Given $\boldsymbol{u}^n$, *compute* $\boldsymbol{u} = \boldsymbol{u}^{n+1}$ *and* $p = p^{n+1}$ *by solving*

$$\begin{aligned}[\alpha \mathbf{I} + \theta(\boldsymbol{u} \cdot \nabla - \nu \triangle]\, \boldsymbol{u} + \nabla p = [\alpha I - \theta_1(\boldsymbol{u}^n \cdot \nabla - \nu \triangle]\, \boldsymbol{u}^n \\ + \theta_2 \boldsymbol{f}^{n+1} + \theta_3 \boldsymbol{f}^n\end{aligned} \tag{2}$$

subject to the incompressibility constraint $\nabla \cdot \boldsymbol{u} = 0$
Here, $(\cdot)^n$ indicates the value of the generic quantity $(\cdot)$ at time step t_n for

time-dependent problems or the n-th iteration for the steady-state formulation. The time-dependent problem is defined for $\alpha = 1/\Delta t$, while the steady-state formulation is recovered for $\alpha = 0$, $\theta = \theta_1 = \theta_3 = 1$, and $\theta_2 = 0$.

For the spatial discretization let V_h and Q_h be approximative spaces of $H_0^1(\Omega)$, and $L_2(\Omega)$ respectively, then the resulting discrete problems have the following algebraic form:
Compute $\boldsymbol{u}$ *and* p *by solving*

$$\mathbf{A}\boldsymbol{u} + \mathbf{B}\,p = \boldsymbol{g} \quad , \quad \mathbf{B}^T\boldsymbol{u} = 0 \qquad \text{where} \tag{3}$$

$$\boldsymbol{g} = [\alpha M - \theta_1 \mathbf{L} - \theta_1 N(\boldsymbol{u}^n)]\,\boldsymbol{u}^n + \theta_2 f^{n+1} + \theta_3 f^n \tag{4}$$

Here, M is the (consistent or lumped) mass matrix, $\mathbf{B}$ is the discrete gradient operator and $-\mathbf{B}^T$ is the associated divergence operator. Furthermore,

$$\mathbf{A}\boldsymbol{u} = [\alpha M + \theta \mathbf{L} + \theta \mathbf{N}(\boldsymbol{u})]\,\boldsymbol{u}, \tag{5}$$

where $\mathbf{L}$ is the viscous term and $\mathbf{N}(\boldsymbol{u})$ is the nonlinear transport operator. Furthermore, the discretized equations (2) as well as the linear subproblems can be solved within the outer iteration loop by a fixpoint defect correction or Newton method.

In this paper, we employ the stable $\tilde{Q}_1/Q_0$ finite element pair. In the two-dimensional case, the nodal values are the mean or midpoint values of the velocity vector over the element edges, and the mean values of the pressure over the elements (see [2]). There are two well-known situations for nonconforming finite element methods when severe numerical problems may arise: Firstly, the lack of coercivity for nonconforming low order approximations for symmetric deformation tensor formulations, mainly visible for small Re numbers. Secondly, convection dominated problems, for instance for medium and high Re numbers or for the treatment of pure transport problems. Then, the standard Galerkin formulation fails and may lead to numerical oscillations or convergence problems of the iterative solvers, too (see[1, 4]).

Among the stabilization methods existing in the literature for these types of problems, we use the proposed one in [4] which is based on the penalization of the gradient jumps over element boundaries. It takes the following form (with $h_E = |E|$)

$$\langle \mathbf{J}\boldsymbol{u}, \boldsymbol{v}\rangle = \sum_{\text{edge E}} \max(\gamma^* \nu h_E, \gamma h_E^2) \int_E [\nabla \boldsymbol{u}] : [\nabla \boldsymbol{v}]\, d\sigma, \tag{6}$$

and will be added to the original bilinear form in order to cure numerical instabilities when computing incompressible flow problems using low order nonconforming finite elements. Moreover, only one generic stabilization term takes care of all instabilities (see [4]).

2 Sparsity of the Matrix

Sparse matrices are an integral part of the FEM analysis for incompressible flow problems which may lead to huge and ill-conditioned systems so that very fast solvers of Krylov-space or particularly of multigrid type are required. In addition the introduced edge-oriented stabilization techniques destroy the typical local sparsity properties since this approach involves more than the adjacent elements: The corresponding rows and columns for the new stiffness matrices $\mathbf{J}$ may contain 23 nonzero matrix elements, in contrast to the usual 7 for the non-stabilized case in 2D (see Fig. 1), and 61 nonzero matrix elements in contrast to the usual 11 for the non-stabilized case in 3D.

2.1 Storage in the same FEM data structure

To overcome the problem of storing the new matrix $\mathbf{J}$ – coming from $\langle \mathbf{J}\boldsymbol{u}, \boldsymbol{v}\rangle$ – with regard to the standard FEM data structures, the matrix $\mathbf{J}$ is written as a sum of two matrices $\mathbf{J}^*$ and $\mathbf{J}_{rest}$,

$$\mathbf{J} = \mathbf{J}^* + \mathbf{J}_{rest} \tag{7}$$

where $\mathbf{J}^*$ has the same sparsity structure as the usual corresponding finite element matrix; then, $\mathbf{J}_{rest} = \mathbf{J} - \mathbf{J}^*$. Hence, $\mathbf{J}^*$ can be handled with the same linear algebra techniques which are typically used for the treatment of the standard nonconforming finite element approach; $\mathbf{J}_{rest}$ is the complementary part and will be used as a correction for the calculation of the residuals inside of the linear solvers only. Then, given any approximation $\boldsymbol{v}$, and by $\mathbf{A}$ denoting the standard stiffness matrix from (5) without the new stabilization matrices, we can write the complete residual as:

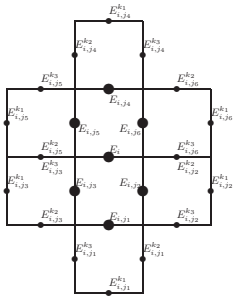
$$\boldsymbol{f} - (\mathbf{A} + \mathbf{J})\boldsymbol{v} = \boldsymbol{f} - (\mathbf{A} + \mathbf{J}^*)\boldsymbol{v} - \mathbf{J}_{rest}\boldsymbol{v} \tag{8}$$

Consequently, only the partial matrix $\mathbf{A} + \mathbf{J}^*$ has to be stored in the complete stiffness matrix so that the first part of the residual can be obtained via standard matrix-vector multiplication while the second part is assembled via elementwise operations. Moreover, the construction of preconditioners for the corresponding linear systems may only include parts of the (sub)matrix $\mathbf{A}+\mathbf{J}^*$, too, which will be explained in the following (see also [3] for more details).

2.2 Storage in a special edge-oriented data structure

A data structure for the storage of the stiffness matrix for edge-oriented stabilization is not common in FEM community. Fortunately, it is not difficult to develop one from the available FEM storage techniques. In fact, each edge E_i has two surrounding elements with n_i edges $(E_{i,j})_{j=1}^{n_i}$, then by the intermediate of the edges $(E_{i,j})_{j=1}^{n_i}$ the other contributed elements and edges $(E_{i,j_k})_{k=1}^{m_j}$ required for edge-oriented stabilization techniques are obtained (see Fig. 1).

This is exactly the graph of the extended matrix: In fact, let the index i be assimilated to any matrix row and the index j be the corresponding nonzero columns in the standard FEM data structure, the extension will consist of the corresponding nonzero columns j_k to the rows j.



Storage Technique	two-dimensional mesh			three-dimensional mesh		
		FEM	EO		FEM	EO
Level	Elements	Matrix entries		Elements	Matrix entries	
1	4	60	128	27	918	3474
2	16	232	628	216	7236	32436
3	64	912	2732	1728	57456	277632
4	246	3616	11356	13824	457920	2292768
5	1024	14400	46268	110592	3656448	18627264
6	4096	57472	186748	884736	29223936	150155136

Fig. 1. An illustration for edge-oriented storage technique and the total number of nonzero matrix entries for the $\tilde{Q}_1$ element on a unit square.

Edge-oriented storage algorithm

Based on the standard Compressed Sparse Row CSR-FEM storage technique, let N_A be the number of entries in the matrix A, N_{Eq} be the number of equations, $P_c(N_A)$ a vector with dimension N_A to be the column pointer and $P_r(N_{Eq}+1)$ a vector with dimension $N_{Eq}+1$ to be the pointer row. Then, the edge-oriented storage technique is deduced from the standard FEM storage technique as following

$$\tilde{N}_A = 1 \quad , \quad l_1 \quad = 1. \tag{9}$$

For each $i = 1, .., N_{eq}$ the corresponding nonzero columns are given by the following nested loops

$$\tilde{P}_r(i) = l_i. \tag{10}$$

1. In standard FEM storage we get

$$i_j = P_c(l), \quad P_r(i) \le l \le P_r(i+1) - 1. \tag{11}$$

2. For each i_j the extension consists of the nonzero corresponding column in the standard FEM storage which is given by

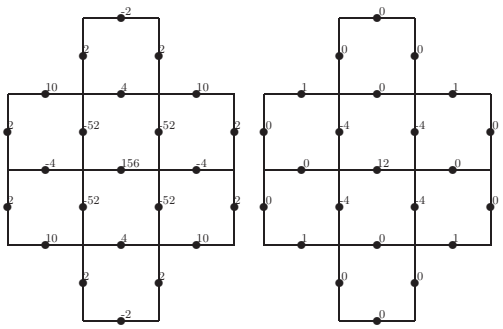
$$\begin{aligned} k_{i_j} &= P_c(l), \quad P_r(i_j) \le l \le P_r(i_j+1) - 1 \\ \tilde{N}_A &= \tilde{N}_A + 1; \quad l_i = l_i + 1; \quad \tilde{P}_c(l_i) = k_{i_j}. \end{aligned} \tag{12}$$

Here, $\tilde{N}_A$ denotes the number of entries in the matrix A, $\tilde{P}_r$ is the row pointer and $\tilde{P}_c$ is the column pointer in the edge-oriented storage. In practice we consider the so-called ***edge-oriented patches*** Ω_i which consist of the neighboring elements sharing the same edge

$$\Omega_i = \cup \{T, T \in \mathcal{T}_h \wedge \cap_{T \in \mathcal{T}_h} = E_i\}. \tag{13}$$

All our elementary operations will be based on Ω_i.

Looking more carefully at the resulting matrix stencils for the terms $\int_E [\nabla\phi_i][\nabla\phi_j]d\sigma$, the matrix structure can be seen in Fig. 2. While the matrix stencils are always increased, leading to couplings between FEM basis functions which do not have common local support, it is also visible that reduced integration, for instance via midpoint rule, may lead to a different amount of additional memory requirements. We can see this reduction for the $\tilde{Q}_1$ ele-



Level	Quadrature technique exact Gauss	1x1 Gauss
1	128	76
2	628	328
3	2732	1360
4	11356	5536
5	46268	22336
6	186748	89728

Fig. 2. Stencil for $\int_E [\nabla\phi_i][\nabla\phi_j]d\sigma$ with exact (left), and with 1x1 Gauss quadrature (middle); total number of nonzero matrix entries (right) for the $\tilde{Q}_1$ element with midpoints as degree of freedom on the unit square.

ment with midpoint values on edges as degree of freedom, which shows that the connections for the edge-oriented finite element methods can be chosen optimally which will lead to moderately increased matrix stencils. A more detailed analysis will be performed in a forthcoming paper.

3 Local Pressure Schur Complement Approach

Local Pressure Schur complement schemes (see [2]) as generalization of so-called Vanka smoothers are simple iterative methods for coupled systems

$$\begin{pmatrix} \mathbf{A}+\mathbf{J} & \mathbf{B} \\ \mathbf{B}^T & 0 \end{pmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{Res}_u \\ \mathbf{Res}_p \end{bmatrix}, \tag{14}$$

of saddle point type which are acting directly on element level and which are embedded into an outer block Jacobi/Gauss-Seidel iteration. The local character of this procedure together with a global defect-correction mechanism is crucial for our approach. If $\mathbf{Res}_u$ and $\mathbf{Res}_p$ denote the residuals for the (complete) discrete momentum and continuity equations which include the complete stabilization term due to $\mathbf{J}$ as described in (6), then, two types of Vanka smoothers can be applied with respect to the decomposition of the domain Ω to patches $\{\Omega_i, i = 1, ..., I\}$ which is not required to be disjoined.

3.1 Cell-oriented Vanka smoother

In this case the *patches* Ω_i may consist of only one element and the index I is the total number of elements, which means that the global stiffness matrix is restricted to the single cells/quadrilaterals of the mesh. It is straightforward to deduce the element stiffness matrix from the global stiffness matrix as follows

$$K = \sum_{T \in \mathcal{T}_h} K_T, \tag{15}$$

where K and K_T denote the global and element stiffness matrices respectively. It follows that

$$\begin{aligned} [K_T]_{ij} &= \left[\mathbf{A}_{|T}\right]_{ij} + \left[\mathbf{J}_{|T}\right]_{ij} && \text{for} \quad 1 \le i,j \le 4 \\ [K_T]_{i5} &= \left[\mathbf{B}_{|T}\right]_{i} && \text{for} \quad 1 \le i \le 4 \\ [K_T]_{5i} &= \left[\mathbf{B}^T_{|T}\right]_{i} && \text{for} \quad 1 \le i \le 4 \\ [K_T]_{55} &= 0. \end{aligned} \tag{16}$$

With the standard FEM data structure (without the extension of the matrix) the contribution of the jump term will be restricted to $\mathbf{J}^*$. Then, there holds

$$[K^*_T]_{ij} = \left[\mathbf{A}_{|T}\right]_{ij} + \left[\mathbf{J}^*_{|T}\right]_{ij} \quad \text{for} \quad 1 \le i,j \le 4. \tag{17}$$

The basic iteration step, which means one smoothing step, can be described as follows

$$\begin{bmatrix} \boldsymbol{u}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^{n} \\ p^{n} \end{bmatrix} + \omega^n \sum_{T \in \mathcal{T}_h} \left[\tilde{K}^*_T\right]^{-1} \begin{bmatrix} \mathbf{Res}_u(u^n, p^n) \\ \mathbf{Res}_p(u^n, p^n) \end{bmatrix}_{|T} \tag{18}$$

where the matrix $\tilde{K}^*_T$ is easily invertible and remains close to K^*_T. Related to the choice of the matrix $\tilde{K}^*_T$ two types of Vanka smoothers are described, namely diagonal Vanka smoother and full Vanka smoother.

(a) *Diagonal Vanka smoother:* The diagonal Vanka smoother updates the velocity and the pressure values connected to the element T by

$$\begin{bmatrix} \boldsymbol{u}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^{n} \\ p^{n} \end{bmatrix} + \omega^n \sum_{T \in \mathcal{T}_h} \left[\mathrm{diag}(K^*_T)\right]^{-1} \begin{bmatrix} \mathbf{Res}_u(u^n, p^n) \\ \mathbf{Res}_p(u^n, p^n) \end{bmatrix}_{|T}. \tag{19}$$

(b) *Full Vanka smoother:* The full Vanka smoother updates the velocity and the pressure values connected to the element T by

$$\begin{bmatrix} \boldsymbol{u}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^{n} \\ p^{n} \end{bmatrix} + \omega^n \sum_{T \in \mathcal{T}_h} \left[K^*_T\right]^{-1} \begin{bmatrix} \mathbf{Res}_u(u^n, p^n) \\ \mathbf{Res}_p(u^n, p^n) \end{bmatrix}_{|T}. \tag{20}$$

As can be seen, for the preconditioning step only parts of the matrix (here: $\mathbf{A} + \mathbf{J}^*$) are involved while the residual contains all parts of the matrix. Consequently, when this approach converges, the result is the solution of the stabilized version while the preconditioning steps only determine the speed of the overall iteration procedure.

3.2 Edge-oriented Vanka smoother

To incorporate the full jump $\mathbf{J}$ into the preconditioning step we use the edge-oriented patches Ω_i. This will keep the size of the local problem small and the full matrix $\mathbf{J}$ will be used for the preconditioning steps. The extension of the matrix to support the jump term leads to a 5×5 FEM matrix block of the type (16). To keep the size of the local problem small, the element matrix is disassembled to its edge contributions

$$K_T = \sum_{i=1}^{m} K_T^{E_i}, \tag{21}$$

where $K_T^{E_i}$ is the contribution of the edge E_i to K_T and m is the number of the edges on the cell T. From the definition of edge-oriented patches (13), the edge stiffness matrix may contain the contributions of all sharing elements

$$K^{E_i} = \sum_{T \in \Omega_i} K_T^{E_i} = K_{\Omega_i}^{E_i}. \tag{22}$$

Then, one basic iteration can be described as follows

$$\begin{bmatrix} \boldsymbol{u}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^{n} \\ p^{n} \end{bmatrix} + \omega^n \sum_{i \in I} \left[K_{\Omega_i}^{E_i} \right]^{-1} \begin{bmatrix} \mathbf{Res}_u(u^n, p^n) \\ \mathbf{Res}_p(u^n, p^n) \end{bmatrix}_{|\Omega_i}, \tag{23}$$

where I is the total number of internal edges. This blocking strategy is different from that used in [2] to generate isotropic subdomains for stabilizing strong mesh anisotropy. Indeed, for the edge-oriented patches the number of block matrices is only depending on the number of edges and not on the number of patches itself.

The global defect restricted to a single patch Ω_i is given by

$$\begin{bmatrix} \mathbf{Res}_u(u^n, p^n) \\ \mathbf{Res}_p(u^n, p^n) \end{bmatrix}_{|\Omega_i} = \left(\begin{bmatrix} \mathbf{L} + \tilde{\mathbf{N}} + \mathbf{J} & \mathbf{B} \\ \mathbf{B}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^n \\ p^n \end{bmatrix} - \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix} \right)_{|\Omega_i}. \tag{24}$$

In practice the following auxiliary problem

$$\left[K_{\Omega_i}^{E_i} \right] \begin{bmatrix} \boldsymbol{v}_i^{n+1} \\ q_i^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{Res}_u(u^n, p^n) \\ \mathbf{Res}_p(u^n, p^n) \end{bmatrix}_{|\Omega_i} \tag{25}$$

is solved, and then the new iterates $\boldsymbol{u}^{n+1}$ and p^{n+1} are computed

$$\begin{bmatrix} \boldsymbol{u}^{n+1} \\ p^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^{n} \\ p^{n} \end{bmatrix} + \omega^n \sum_{i \in I} \begin{bmatrix} \boldsymbol{v}_i^{n+1} \\ q_i^{n+1} \end{bmatrix}. \tag{26}$$

The resulting local MPSC method corresponds to a simple block-Jacobi iteration for the mixed problem (14) and to a block-Gauss-Seidel method by using the updated solution for the computation of the local defect (24).

4 Numerical Example

The realistic evaluation of the efficiency of the edge-oriented FEM storage technique versus the standard one is difficult to handle because of the interplay of different components. Here, we restrain the numerical examples to the DFG benchmark of flow around cylinder (see [4]).

Our numerical test (Stokes problem) is concerned with the symmetric deformation tensor formulation to show the advantage of using the edge-oriented stabilization with special storage technique. We also present the gradient formulation for comparison since it does not require any stabilization. In Table 1, we list the total number of multigrid sweeps (MG) and the total CPU time for both storage techniques with and without using the jump terms.

Table 1. Vanka smoother coupled with standard and edge-oriented FEM for the symmetric deformation tensor and gradient formulations

	edge-oriented storage technique				standard FEM storage technique			
	without jump stab.		with jump stab.		without jump stab.		with jump stab.	
Level	MG	Time	MG	Time	MG	Time	MG	Time
				the gradient formulation				
4	12	37	12	44	12	32	12	180
5	12	153	11	166	12	128	12	780
6	12	634	11	676	12	531	11	2594
				the deformation tensor formulation				
4	191	542	8	28	191	442	8	115
5	535	6209	9	133	535	5426	9	524
6	1225	63614	8	525	1225	49502	8	1905

The results in Table 1 for several mesh refinement levels show that the edge-oriented storage technique moderately increases the CPU cost. Moreover, the need for edge-oriented stabilization for the deformation tensor formulation is cleary visible.

Summarising, we have developed new techniques to make edge-oriented FEM stabilizations more advantageous for CFD simulations. However, more research is required concerning the corresponding time-accurate methods and approximate preconditioners for global Pressure Schur Complement schemes.

References

1. Burman, E. and Hansbo, P.: A stabilized non-conforming finite element method for incompressible flow, J. Comput. Methods. Appl. Mech. Engrg. (2004) accepted
2. Turek, S.: Efficient solvers for incompressible flow problems: An algorithmic and computational approach. Springer, Berlin-Heidelberg (1999)
3. Turek, S., Ouazzi, A. and Schmachtel, R.: Multigrid method for stabilized non-conforming finite elements for incompressible flow involving the deformation tensor formulation, JNM, **10**, 235–248 (2002)
4. Turek, S. and Ouazzi, A.: Unified edge–oriented stabilization of nonconforming finite element methods for incompressible flow problems: Numerical investigations, JNM, (2005) (accepted)