Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

An efficient direct solver for the boundary concentrated FEM in 2D

by

B. N. Khoromskij and J. M. Melenk

Preprint no.: 64 2001

An Efficient Direct Solver for the Boundary Concentrated FEM in 2D

B.N. Khoromskij and J.M. Melenk, Leipzig

Abstract

The boundary concentrated FEM, a variant of the *hp*-version of the finite element method, is proposed for the numerical treatment of elliptic boundary value problems. It is particularly suited for equations with smooth coefficients and boundary conditions that may have low Sobolev regularity. In the two-dimensional case, it is shown that the Cholesky factorization of the resulting stiffness matrix requires $O(N \log^4 N)$ units of storage and can be computed with $O(N \log^8 N)$ work. Numerical results confirm theoretical estimates.

Keywords: *hp*-finite element methods, meshes refined towards boundary, direct solvers, Schur complement
AMS Subject Classification: 65N35, 65F10, 35D10

1 Introduction

The recently introduced *boundary concentrated finite element method* of [8] is a numerical method that is particularly suited for solving elliptic boundary value problems with the following two properties: a) the coefficients of the equations are analytic so that, by elliptic regularity, the solution is analytic; b) globally, the solution has low Sobolev regularity due to, for example, boundary conditions with low regularity or non-smooth geometries. The boundary concentrated FEM exploits interior regularity in the framework of the *hp*-version of the finite element method (*hp*-FEM) by using special types of meshes and polynomial degree distributions: meshes that are strongly refined toward the boundary (see Figs. 5, 6 for typical examples) are employed in order to cope with the limited regularity near the boundary; away from the boundary where the solution is smooth, high approximation order is used on large elements. In fact, judiciously linking the approximation order to the element size leads to optimal approximation results (see Theorem 2.4 and Remark 2.6 for the precise notion of optimality).
In the present paper, we focus on the boundary concentrated FEM in two space dimensions and present a scheme for the Cholesky factorization of the resulting stiffness matrix that requires $O(N \log^4 N)$ units of storage and $O(N \log^8 N)$ work; here, N is the problem size. The key to this efficient Cholesky factorization scheme is an algorithm that numbers the unknowns such that the profile of the stiffness matrix is very small (see Fig. 1 for a typical sparsity pattern of the Cholesky factor). Numerical examples confirm our complexity estimates.
The boundary concentrated FEM can be used to realize a fast (i.e., with linear-logarithmic complexity) application of discrete Poincaré-Steklov and Steklov-Poincaré operators as we will discuss in Section 2.5. This use of the boundary concentrated FEM links it to the classical

boundary element method (BEM). Indeed, it may be regarded as a generalization of the BEM: While the BEM is effectively restricted to equations with constant coefficients, the boundary concentrated FEM is applicable to equations with variable coefficients yet retains the rate of convergence of the BEM. Since the Cholesky factorization of the stiffness matrix allows for an exact, explicit data-sparse representation of boundary operators such as the Poincaré-Steklov operator with linear-logarithmic complexity, the boundary concentrated FEM provides a sparse direct solver for the 2D BEM (because it directly computes the full set of Cauchy data on the boundary corresponding to $\mathcal{L}$-harmonic functions) that has almost linear complexity. It is also a new alternative to modern matrix compression techniques now used in BEM.
We present a Cholesky factorization scheme for the boundary concentrated FEM in two dimensions, that is, a direct method. We mention that iterative methods for solving the system of linear equations arising in the boundary concentrated FEM are considered in [8]. Depending on the boundary conditions considered, different preconditioners are required for an efficient iterative solution method. For example, while for Dirichlet problems the condition number of the stiffness matrix grows only polylogarithmically with the problem size, [8], Neumann problems require more effective preconditioning. A suitable preconditioner, which has block-diagonal structure, was proposed in [8]. We point out, however, that an application of this preconditioner requires an inner iteration making our direct solver an attractive option.
We conclude this introduction by mentioning that, although we restrict our exposition to the case of symmetric positive definite problems, our procedure can be extended to non-symmetric problems by constructing the LU-decomposition rather than the Cholesky factorization.

2 Boundary Concentrated FEM

In this section, we present a brief survey of the boundary concentrated FEM. We refer to [8] for a detailed description of this technique and complete proofs. For the sake of concreteness, we discuss Dirichlet problems although other types of boundary conditions such as Neumann or mixed boundary conditions can be treated analogously.

2.1 Problem class and abstract Galerkin FEM

For a *polygonal* Lipschitz domain $\Omega \subset \mathbb{R}^2$, we consider the Dirichlet problem

$$\mathcal{L}u = f \in L^2(\Omega), \qquad \text{in } \Omega, \tag{2.1a}$$
$$\gamma_0 u = \lambda \in H^{1/2}(\partial\Omega) \qquad \text{on } \partial\Omega, \tag{2.1b}$$

where the differential operator $\mathcal{L}$ is given by

$$\mathcal{L}u := -\nabla \cdot (A\nabla u) + a_0 u \tag{2.2}$$

with uniformly (in $x \in \overline{\Omega}$) symmetric positive definite matrix $A = (a_{ij})_{i,j=1}^2$. Moreover, in the boundary concentrated FEM, we assume that A and the scalar functions a_0, f are analytic on $\overline{\Omega}$. The operator $\gamma_0 : H^1(\Omega) \to H^{1/2}(\partial\Omega)$ is the trace operator that restricts functions on Ω to the boundary $\partial\Omega$. We assume that the operator $\mathcal{L}$ generates an $H^1(\Omega)$-elliptic bilinear form

$$B(u,v) = \int_\Omega \sum_{i,j=1}^2 a_{ij}\partial_j u \partial_i v + a_0 uv \, dx, \tag{2.3}$$

i.e.,

$$c_0 \|u\|_{1,\Omega}^2 \leq B(u,u) \leq c_1 \|u\|_{1,\Omega}^2 \qquad \forall u \in H_0^1(\Omega). \tag{2.4}$$

The boundary value problem (2.1) is understood in the usual, variational sense. That is, solving (2.1) is equivalent to the problem:

$$\text{Find } u \in H^1(\Omega) \text{ with } \gamma_0 u = \lambda \text{ and } B(u,v) = \int_\Omega f\, v\, dx \qquad \forall v \in H_0^1(\Omega). \tag{2.5}$$

The standard FEM is obtained from the weak formulation (2.5) by replacing the space $H^1(\Omega)$ with a finite dimensional subspace $V_N \subset H^1(\Omega)$. For the Dirichlet problem (2.1), we introduce the trace space

$$Y_N := V_N|_{\partial\Omega} = \{\gamma_0 v \mid v \in V_N\} \subset H^{1/2}(\partial\Omega). \tag{2.6}$$

For an approximation $\lambda_N \in Y_N$ to λ we can then define the FEM for (2.5) as follows:

$$\text{Find } u_N \in V_N \text{ s.t. } u_N = \lambda_N \quad \text{ and } \quad B(u_N, v) = \int_\Omega f\, v\, dx \qquad \forall v \in V_N \cap H_0^1(\Omega). \tag{2.7}$$

The coercivity assumption (2.4) ensures existence of the finite element approximation u_N. Furthermore, by Céa's Lemma there is a $C > 0$ independent of V_N such that u_N satisfies

$$\|u - u_N\|_{H^1(\Omega)} \;\leq\; C \inf_{\substack{v \in V_N \\ \gamma_0 v = \lambda_N}} \left\{ \|u - v\|_{H^1(\Omega)} + \|\lambda_N - \lambda\|_{H^{1/2}(\partial\Omega)} \right\}. \tag{2.8}$$

In practice, the approximations λ_N are obtained with the aid of the L^2-projection operator $Q_N : H^{1/2}(\partial\Omega) \to Y_N$ by setting $\lambda_N = Q_N\lambda$, i.e., for $\lambda \in L^2(\partial\Omega)$ the function $Q_N\lambda$ is defined by

$$\langle Q_N \lambda, v\rangle_{0,\partial\Omega} = \langle \lambda, v\rangle_{0,\partial\Omega} \qquad \forall v \in Y_N. \tag{2.9}$$

In the next section, we specify the approximation spaces V_N, the proper choice of which is intimately linked to the regularity properties of the solution u of (2.1). The analyticity of the data A, a_0, and f implies by interior regularity that the solution u is analytic on Ω; if furthermore $u \in H^{1+\delta}(\Omega)$ for some $\delta \in (0,1]$, then the blow-up of higher order derivatives near the boundary can be characterized precisely in terms of so-called countably normed spaces (see [8] for the details). This regularity allows us to prove an optimal error estimate for the boundary concentrated FEM in Theorem 2.4 below.

2.2 Geometric meshes and linear degree vectors

For ease of exposition, we will restrict our attention to regular triangulations (i.e., no hanging nodes) consisting of *affine triangles.* (We refer, for example, to [?] for the precise definition of regular triangulations.) We emphasize, however, that an extension to quadrilateral and curvilinear elements is possible. The triangulation $\mathcal{T} = \{K\}$ of the domain Ω consists of elements K, each of which is the image $F_K(\hat{K})$ of the equilateral reference triangle

$$\hat{K} = \left\{ (x,y) \mid -1 < x < 1,\ 0 < y < \sqrt{3}(1 - |x|) \right\}$$

under the *affine* map F_K. We furthermore assume that the triangulation $\mathcal{T}$ is γ-*shape-regular*, i.e.,

$$h_K^{-1} \|F_K'\|_{L^\infty(\hat{K})} + h_K \|(F_K')^{-1}\|_{L^\infty(\hat{K})} \leq \gamma \qquad \forall K \in \mathcal{T}. \tag{2.10}$$

Here, h_K denotes the diameter of the element K. Of particular importance to us will be *geometric meshes*, which are strongly refined meshes near the boundary $\partial\Omega$:

Definition 2.1 (geometric mesh) *A γ-shape-regular (cf. (2.10)) mesh $\mathcal{T}$ is called a* geometric mesh *with* boundary mesh size h *if there exist c_1, $c_2 > 0$ such that for all $K \in \mathcal{T}$:*

1. *if $\overline{K} \cap \partial\Omega \neq \emptyset$, then $h \leq h_K \leq c_2 h$;*
2. *if $\overline{K} \cap \partial\Omega = \emptyset$, then $c_1 \inf_{x\in K} \operatorname{dist}(x, \partial\Omega) \leq h_K \leq c_2 \sup_{x\in K} \operatorname{dist}(x, \partial\Omega)$.*

Typical examples of geometric meshes are depicted in Figs. 5, 6. Note that the restriction to the boundary $\partial\Omega$ of a geometric mesh is a quasi-uniform mesh, which justifies speaking of a "boundary mesh size h".
In order to define *hp*-FEM spaces on a mesh $\mathcal{T}$, we associate a polynomial degree $p_K \in \mathbb{N}$ with each element K, collect these p_K in the polynomial degree vector $\mathbf{p} := (p_K)_{K\in\mathcal{T}}$ and set

$$S^{\mathbf{p}}(\Omega, \mathcal{T}) := \{u \in H^1(\Omega) \,|\, u \circ F_K \in \mathcal{P}_{p_K}(\hat{K}) \quad \forall K \in \mathcal{T}\}, \tag{2.11}$$

$$S^{\mathbf{p}}_0(\Omega, \mathcal{T}) := S^{\mathbf{p}}(\Omega, \mathcal{T}) \cap H^1_0(\Omega), \tag{2.12}$$

where for $p \in \mathbb{N}$ we introduce the space of all polynomials of degree p as

$$\mathcal{P}_p(\hat{K}) = \operatorname{span}\{x^i y^j \,|\, 0 \leq i + j \leq p\}.$$

The *linear degree vector* is a particularly useful polynomial degree distribution in conjunction with geometric meshes:

Definition 2.2 (linear degree vector) *Let $\mathcal{T}$ be a geometric mesh with boundary mesh size h in the sense of Definition 2.1. A polynomial degree vector $\mathbf{p} = (p_K)_{K\in\mathcal{T}}$ is said to be a* linear degree vector with slope $\alpha > 0$ *if*

$$1 + \alpha c_1 \log \frac{h_K}{h} \leq p_K \leq 1 + \alpha c_2 \log \frac{h_K}{h}.$$

An important observation about geometric meshes and linear degree vectors is that the dimension $\dim S^{\mathbf{p}}(\Omega, \mathcal{T})$ of the space $S^{\mathbf{p}}(\Omega, \mathcal{T})$ is proportional to the number of points $N_\Gamma = O(h^{-1})$ on the boundary:

Proposition 2.3 ([8]) *Let $\mathcal{T}$ be a geometric mesh with boundary mesh size h. Let $\mathbf{p}$ be a linear degree vector with slope $\alpha > 0$ on $\mathcal{T}$. Then there exists $C > 0$ depending only on the shape-regularity constant γ and the constants c_1, c_2, α of Definitions 2.1, 2.2 such that*

$$\begin{aligned} \dim S^{\mathbf{p}}(\Omega, \mathcal{T}) &\sim \sum_{K\in\mathcal{T}} p_K^2 \leq C h^{-1}, \\ \max_{K\in\mathcal{T}} p_K &\leq C \log h^{-1}. \end{aligned}$$

2.3 Error and complexity estimates

We formulate an approximation result for the *hp*-FEM on geometric meshes applied to (2.1):

Theorem 2.4 ([8]) *Let u be the solution to (2.1) with coefficients A, a_0, and right-hand side f analytic on $\overline{\Omega}$. Assume additionally that $u \in H^{1+\delta}(\Omega)$ for some $\delta \in (0, 1)$. Let $\mathcal{T}$ be a geometric mesh with boundary mesh size h and let $\mathbf{p}$ be a linear degree vector on $\mathcal{T}$ with slope $\alpha > 0$. Then the FE solution u_N given by (2.7) satisfies*

$$\|u - u_N\|_{H^1(\Omega)} \leq C \left[h^\delta + h^{b\alpha}\right]. \tag{2.13}$$

The constants C, $b > 0$ depend only on the shape-regularity constant γ, the constants c_1, c_2 appearing in Definition 2.1, the data A, c, f, Ω, and δ, $\|u\|_{H^{1+\delta}(\Omega)}$. For α sufficiently large the boundary concentrated FEM achieves the optimal rate of convergence

$$\|u - u_N\|_{H^1(\Omega)} \le Ch^\delta = O(N_\Gamma^{-\delta}), \qquad N_\Gamma = \textit{number of boundary points.}$$

Remark 2.5 Theorem 2.4 is formulated for the Dirichlet problem (2.1). Analogous approximation results hold for Neumann or mixed boundary conditions as well. ▪

Remark 2.6 Theorem 2.4 asserts a rate $O(n^{-\delta})$ for the boundary concentrated FEM, where $n = \dim S^{\mathbf{p}}(\Omega, \mathcal{T}) \sim N_\Gamma$. This rate is optimal in the following sense: Setting for $\delta \in (0,1)$

$$\mathcal{U}_\delta := \{u \in H^{1+\delta}(\Omega) \mid \|u\|_{H^{1+\delta}(\Omega)} \le 1 \text{ and } u \text{ solves } \mathcal{L}u = 0 \text{ on } \Omega\}$$

we can introduce the n-width

$$d_n := \inf_{E_n} \sup_{u \in \mathcal{U}_\delta} \inf_{v \in E_n} \|u - v\|_{H^1(\Omega)}, \tag{2.14}$$

where the first infimum is taken over all subspaces $E_n \subset H^1(\Omega)$ of dimension n. It can then be shown that $Cn^{-\delta} \le d_n$ for some $C > 0$ independent of n. ▪

2.4 Shape functions and stiffness matrix

In order to convert the variational formulation (2.7) into a system of linear equations, a basis of the finite element space $S^{\mathbf{p}}(\Omega, \mathcal{T})$ or $S_0^{\mathbf{p}}(\Omega, \mathcal{T})$ has to be chosen. Several choices of basis functions ("shape functions") are standard in *hp*-FEM, [2, 12, 7]. Their common feature is that the shape functions can be associated with the topological entities "vertices," "edges," and "elements" of the triangulation $\mathcal{T}$. This motivates us to introduce the following notion of "standard" bases:

Definition 2.7 *A basis $\mathcal{B}$ of $S^{\mathbf{p}}(\Omega, \mathcal{T})$ is said to be a* standard *hp-FEM basis if each shape function $\varphi \in \mathcal{B}$ falls into exactly one of the following three categories:*

1. vertex shape functions*: φ is a vertex shape function associated with vertex V if* $\operatorname{supp}\varphi$ *consists of all elements that have V as a vertex;*

2. edge shape functions*: φ is an edge shape function associated with edge e of $\mathcal{T}$ if* $\operatorname{supp}\varphi$ *consists of the (at most two) elements whose edge includes e;*

3. internal shape functions: *φ is an internal shape function associated with element K if* $\operatorname{supp}\varphi = \overline{K}$.

For a standard basis in this sense, we assign spatial points, called nodes, *to degrees of freedom as follows:*

1. *we assign to the shape function associated with vertex V the point V;*

2. *we assign to the side shape functions associated with edge e the midpoint of e;*

3. *we assign to the internal shape functions associated with element K the barycenter of K.*

One example of a standard basis in the sense of Definition 2.7 is obtained by assembling (see, e.g., [2, 12, 7]) the so-called hierarchical shape functions:

Example 2.8 We construct a basis of the space $S^{\mathbf{p}}(\Omega, \mathcal{T})$ in two steps: In the first step, we define shape functions on the reference element $\hat{K}$. In the second step, we define the basis of $S^{\mathbf{p}}(\Omega, \mathcal{T})$ by an assembling process.
1. step: Define one-dimensional shape functions ϕ_i on the reference interval $(-1, 1)$ by

$$\phi_1(x) = \frac{1}{2}(1+x), \quad \phi_2(x) = \frac{1}{2}(1-x), \quad \phi_i(x) = c_i \int_{\xi=-1}^{x} L_{i-2}(\xi)\, d\xi, \quad i = 3, 4, \dots,$$

where the functions L_i are the standard Legendre polynomials and the scaling factors c_i are given by $c_i = 1/\|L_{i-2}\|_{L^2(-1,1)}$.
Denote by v_i, $i = 1, \dots, 3$, the three vertices of $\hat{K}$ and by Γ_i, $i = 1, \dots, 3$, the three edges (we assume $\Gamma_1 = \{(x, 0) \in \mathbb{R}^2 \mid -1 < x < 1\}$). Let $p_i \in \mathbb{N}$ be polynomial degrees that we associate with the edges Γ_i and let $p \in \mathbb{N}$ be the polynomial degree of the internal shape functions. We then define vertex shape functions $\mathcal{V}$, side shape functions $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$, and internal shape functions $\mathcal{I}$ as follows:

$$\begin{aligned}
\mathcal{V} &:= \text{ the usual linear nodal shape functions } n_i \text{ with } n_i(v_j) = \delta_{ij}, \\
\mathcal{S}_1 &:= \left\{ s_{1,j}(x,y) := \phi_j(x) \frac{y - \sqrt{3}(1+x)}{1+x} \frac{y - \sqrt{3}(1-x)}{1-x} \mid j = 3, \dots, p_1 + 1 \right\}, \\
\mathcal{I} &:= \{ I_{ij}(x,y) := y(y - \sqrt{3}(1+x))(y + \sqrt{3}(1-x)) L_i(x) L_j(y) \mid 0 \le i + j \le p - 3 \}.
\end{aligned}$$

The side shape functions $\mathcal{S}_2$, $\mathcal{S}_3$ are obtained similarly with p_1 replaced with p_2 (resp. p_3) and a suitable coordinate transformation. Note that internal shape functions vanish on $\partial\hat{K}$ and that edge shape functions vanish on two edges.
2. step: Shape functions as defined on the reference element $\hat{K}$ are now assembled to yield a basis of $S^{\mathbf{p}}(\Omega, \mathcal{T})$. First, the standard piecewise linear hat functions are obtained by simply assembling the shape functions $\mathcal{V}$ of each element. The internal shape functions are simply taken as

$$\varphi_{i,j,K}(x,y) = \begin{cases} I_{ij} \circ F_K^{-1}(x,y), & (x,y) \in K \\ 0 & \text{else} \end{cases} \qquad 0 \le i + j \le p_K - 3, \quad K \in \mathcal{T}$$

where $I_{ij} \in \mathcal{I}$ is the internal shape function on the reference element $\hat{K}$ defined above. It remains to assemble the side shape functions. To that end, we associate with each edge e a polynomial degree $p_e := \min\{p_K \mid e \text{ is edge of } K\}$. Let e be an edge shared by two elements K, K'. For simplicity of notation assume that the element maps F_K, $F_{K'}$ are such that $F_K(\Gamma_1) = F_{K'}(\Gamma_1) = e$ and that additionally $F_K(x, 0) = F_{K'}(x, 0)$ for $x \in (-1, 1)$ (we refer to [2, 7] for details on how to treat the general case). We then define $p_e - 1$ edge shape functions $\varphi_{i,e}$ associated with edge e by setting

$$\varphi_{i,e}(x,y) := \begin{cases} s_{1,i} \circ F_K^{-1}(x,y) & (x,y) \in K \\ s_{1,i} \circ F_{K'}^{-1}(x,y) & (x,y) \in K' \\ 0 & \text{else} \end{cases}, \qquad 1 \le i \le p_e - 1.$$

An analogous formula holds for edges e with $e \subset \partial\Omega$. ∎

Once a basis of $V_N = S_0^{\mathbf{p}}(\Omega, \mathcal{T})$ is chosen, the hp-FEM (2.7) can be formulated as seeking the solution $U \in \mathbb{R}^{\dim V_N}$ of a system of linear equations

$$AU = F, \qquad A \in \mathbb{R}^{\dim V_N \times \dim V_N}, \quad F \in \mathbb{R}^{\dim V_N}. \tag{2.15}$$

We mention in passing that computing the stiffness matrix A and the load vector F to sufficient accuracy can be accomplished with work $O(\dim V_N)$, [8].

2.5 Sparse Factorization of the Schur complement

We discuss how the Cholesky factorization of the stiffness matrix A leads to the explicit sparse representation of discrete Poincaré-Steklov operators. Let T be the Poincaré–Steklov operator (Dirichlet-Neumann map)

$$T : \lambda \mapsto \gamma_1 u$$

where $\gamma_1 u$ is the co-normal derivative of the solution u to the equation $\mathcal{D}u = 0$ with boundary condition $\gamma_0 u = \lambda$. Then, with $V_N := S^{\mathbf{p}}(\Omega, \mathcal{T})$ and the trace space $Y_N := \gamma_0 V_N$, the discrete approximation $T_N : Y_N \to Y_N'$ is defined as follows: For $\lambda \in Y_N$, the approximation $T_N \lambda \in Y_N'$ is given by

$$\langle T_N \lambda, v \rangle_{0,\partial\Omega} = B(u_N, \widetilde{v}) \qquad \forall v \in Y_N,$$

where $\widetilde{v} \in V_N$ is an arbitrary extension of v satisfying $\gamma_0 \widetilde{v} = v$, and $u_N \in V_N$ solves

$$\gamma_0 u_N = \lambda \quad \text{and} \quad B(u_N, v) = 0 \quad \forall\, v \in V_N \cap H_0^1(\Omega).$$

An analysis of the error $T - T_N$ was presented in [8]:

Theorem 2.9 *Let Ω be a polygon. Then the following two statements hold:*

1. *There exists $\delta_0 > 1/2$ such that the Poincaré-Steklov operator T maps continuously from $H^{1/2+\delta}(\partial\Omega)$ to $H^{-1/2+\delta}(\partial\Omega)$ for all $\delta \in [0, \delta_0)$, i.e.,*

$$\|Tu\|_{H^{-1/2+\delta}(\partial\Omega)} \le C_\delta \|u\|_{H^{1/2+\delta}(\partial\Omega)} \qquad \forall u \in H^{1/2+\delta}(\partial\Omega). \tag{2.16}$$

2. *Under the hypotheses of Theorem 2.4 (with $\delta \in (0,1)$ as in the statement of Theorem 2.4) there holds for arbitrary $\overline{\delta} \in [0, \delta] \cap [0, \delta_0)$*

$$\|T\lambda - T_N Q_N \lambda\|_{-1/2,\partial\Omega} \le C_{\overline{\delta}} \left[h^{\overline{\delta}} + h^{b\alpha} \right]. \tag{2.17}$$

If a standard basis in the sense of Definition 2.7 of the space $S^{\mathbf{p}}(\Omega, \mathcal{T})$ is chosen, then the shape functions can be split into "interior" and "boundary" shape functions. A shape functions is said to be "interior" if its node (see Definition 2.7) lies in Ω; it is a "boundary" shape function if its node lies on $\partial\Omega$. To this partitioning of basis functions corresponds a block partitioning of the stiffness matrix $A_N \in \mathbb{R}^{\dim V_N \times \dim V_N}$ for the unconstrained space V_N of the following form:

$$A_N := \begin{pmatrix} A_{\Gamma\Gamma} & A_{\Gamma I} \\ A_{I\Gamma} & A_{II} \end{pmatrix}.$$

The subscript I indicates the interior shape functions and Γ marks the boundary shape functions. If we choose $Y_N' = Y_N$, then the matrix representation of the operator T_N is given by the Schur complement

$$\mathbf{T}_N := A_{\Gamma\Gamma} - A_{\Gamma I} A_{II}^{-1} A_{I\Gamma}.$$

Inserting the Cholesky factorization $LL^\top = A_{II}$ leads to the desired direct FE method for the Poincaré-Steklov map

$$\mathbf{T}_N := A_{\Gamma\Gamma} - A_{\Gamma I} L^{-T} L^{-1} A_{I\Gamma}. \tag{2.18}$$

Because the Cholesky factor L can be computed with linear-logarithmic complexity (see Section 3), formula (2.18) provides an efficient representation of the Poincaré-Steklov operator. We finally mention that our factorization scheme for the Schur complement carries over verbatim to the case of the Neumann-Dirichlet map and also to the case of mixed boundary conditions.

3 Cholesky factorization of the stiffness matrix

This section is devoted to the main result of the paper, the development of an efficient Cholesky factorization scheme for the stiffness matrix arising in the 2D-boundary concentrated FEM. The key issue is the appropriate numbering of the degrees of freedom. First, we illustrate this numbering scheme for the case of geometric meshes and constant polynomial degree $p = 1$ (Sections 3.2, 3.3). We start with this simpler case because our numbering scheme for the degrees of freedom (Algorithm 3.10) is based on a binary space partitioning; in the case $p = 1$, the degrees of freedom can immediately be associated with points in space, namely, the vertices of the mesh. The general case of linear degree vectors is considered in Section 3.4.
For the case $p = 1$, we recall that by Definition 2.7, the vertices of the mesh are called nodes. We denote by $\mathcal{V}$ the set of nodes and say that a node V' is a *neighbor* of a node V if there exists an element $K \in \mathcal{T}$ such that V and V' are vertices of K. It will prove useful to introduce the set of neighbors of a node $V \in \mathcal{V}$ as

$$\mathcal{N}(V) := \{V' \in \mathcal{V} \,|\, V' \text{ is a neighbor of } V\}, \tag{3.1}$$

because the sparsity pattern of the stiffness matrix can be characterized with the aid of the sets $\mathcal{N}$.

3.1 Nested dissection in Direct Solvers

Let $A \in \mathbb{R}^{N\times N}$ be a symmetric positive definite matrix. We denote by L its Cholesky factor, i.e., the lower triangular matrix L with $LL^\top = A$. For sparse, symmetric positive matrices $A \in \mathbb{R}^{N\times N}$ it is customary, [5], to introduce the *i-th bandwidth* β_i and the *j-th frontwidth* ω_j of A as

$$\begin{aligned} \beta_i &:= \max\{i - j \,|\, j < i \text{ and } A_{ij} \neq 0\}, \qquad i = 1,\ldots,N, && (3.2)\\ \omega_j &:= |\{k \,|\, k > j \text{ and } A_{kl} \neq 0 \text{ for some } l \leq j\}|, \qquad j = 1,\ldots,N-1. && (3.3) \end{aligned}$$

It can be shown (see Proposition 3.1(i)) that in each row i, only the entries L_{ij} with $i - \beta_i \leq j \leq i$ are non-zero. The j-th frontwidth ω_j measures the number of non-zero subdiagonal entries in column j of L, i.e., $\omega_j = |\{i < j \,|\, L_{ij} \neq 0\}|$.
The *frontwidth* ω is given by

$$\omega := \max_{i=1,\ldots,N} \omega_i. \tag{3.4}$$

The cost of computing the Cholesky factor L can then be quantified in terms of the numbers ω_i:

Proposition 3.1 *Let $A \in \mathbb{R}^{N\times N}$ be symmetric, positive definite. Then*

(i) the storage requirement for L is $\mathrm{nnz} = N + \sum_{i=1}^{N-1} \beta_i = N + \sum_{i=1}^{N-1} \omega_i$;

(ii) the number of floating point operations to compute L is

- *N square roots for the diagonal entries L_{ii},*
- *$\frac{1}{2}\sum_{i=1}^{N-1} \omega_i(\omega_i + 3)$ multiplications.*

In particular, the storage requirement nnz *and the number of floating point operations W can be bounded by*

$$\mathrm{nnz} = O(N\omega), \qquad W = O(N\omega^2). \tag{3.5}$$

Proof. See, for example, [5, Chapters 2, 4]. ∎

In view of the estimate (3.5), various algorithms have been devised to number the unknowns so as to minimize the frontwidth ω; the best-known examples include the Reverse Cuthill-McKee algorithm, [4], the algorithm of Gibbs-Poole-Stockmeyer, [6], and *nested dissection.* For stiffness matrices arising in the 2D-boundary concentrated FEM, we will present an algorithm based on nested dissection in Section 3.3 that numbers the nodes such that $\omega = O(\ln^q N)$ for some $q \in \mathbb{N}_0$.

The basic nested dissection algorithm in FEM reads as follows:

Algorithm 3.2 (nested dissection)

```
nested_dissection($\mathcal{V}, N_0$)
% input: Set of nodes $\mathcal{V}$, starting number $N_0$
% output: numbering of the nodes $\mathcal{V}$ starting with $N_0$
if $|\mathcal{V}| = 1$ then label the element of $\mathcal{V}$ with the number $N_0$
else {
    1. partition the nodes $\mathcal{V}$ into three mutually disjoint sets $\mathcal{V}_{left}$, $\mathcal{V}_{right}$, $\mathcal{V}_{bdy}$ such that
        (a) $|\mathcal{V}_{left}| \approx |\mathcal{V}_{right}|$
        (b) $|\mathcal{V}_{bdy}|$ is "small"
        (c) $\mathcal{N}(V) \subset \mathcal{V}_{left} \cup \mathcal{V}_{bdy}$ for all $V \in \mathcal{V}_{left}$ and $\mathcal{N}(V) \subset \mathcal{V}_{right} \cup \mathcal{V}_{bdy}$ for all $V \in \mathcal{V}_{right}$
    2. if $\mathcal{V}_{left} \neq \emptyset$ call nested_dissection($\mathcal{V}_{left}, N_0$)
    3. if $\mathcal{V}_{right} \neq \emptyset$ call nested_dissection($\mathcal{V}_{right}, N_0 + |\mathcal{V}_{left}|$)
    4. enumerate the elements of $\mathcal{V}_{bdy}$ starting with the number $N_0 + |\mathcal{V}_{left}| + |\mathcal{V}_{right}|$
}
return
```

The key property is (1c). It ensures that the stiffness matrix A has the following block structure:

$$A = \begin{pmatrix} A_{left,left} & 0 & A^\top_{bdy,left} \\ 0 & A_{right,right} & A^\top_{bdy,right} \\ A_{bdy,left} & A_{bdy,right} & A_{bdy,bdy} \end{pmatrix} \tag{3.6}$$

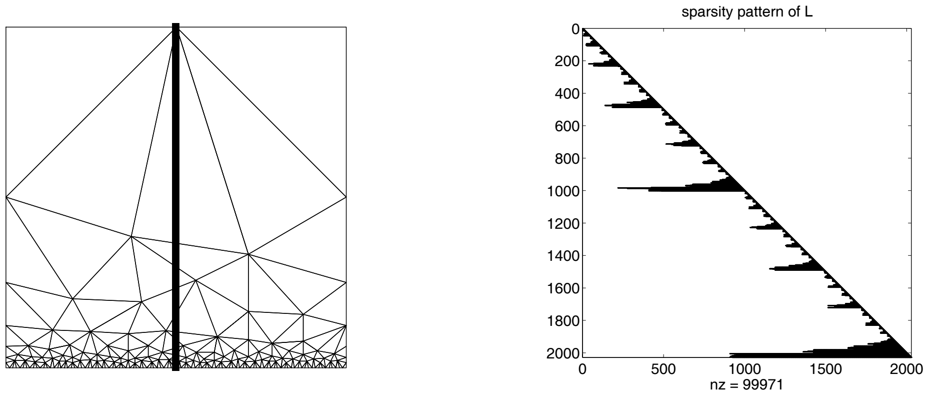

Figure 1: Left: mesh and initial geometric partitioning (thick line). Right: sparsity pattern of Cholesky factor L for mesh with 2028 nodes; $\omega = 98$.

Condition (1b) is imposed in view of the fact that the frontwidth $\omega(A)$ of A can be bounded by

$$\omega(A) \leq \max\{\omega(A_{left,left}), \omega(A_{right,right})\} + |\mathcal{V}_{bdy}|, \tag{3.7}$$

where $\omega(A_{left,left})$, $\omega(A_{right,right})$ are the frontwidths of the submatrices A_{left}, A_{right}. Thus, if the recursion guarantees that $|\mathcal{V}_{bdy}|$ is small and the frontwidths of these submatrices are small, then the numbering scheme is efficient. Since nested dissection operates recursively on the sets $\mathcal{V}_{left}$, $\mathcal{V}_{right}$, its effectiveness hinges on the availability of partitioning strategies for Step 1 of Algorithm 3.2 that yield small $|\mathcal{V}_{bdy}|$. The particular node distributions appearing in the boundary concentrated FEM will allow us to devise such a scheme in Section 3.3.

3.2 Nested dissection: an Example

We show that for meshes that arise in the boundary concentrated FEM, it is possible to perform the partitioning of Step 1 in Algorithm 3.2 such that the set $\mathcal{V}_{bdy}$ is very small compared to $\mathcal{V}$. We illustrate this in the following example.

Example 3.3 Consider meshes that are refined toward a single edge as shown in Fig. 1. The thick line partitions $\mathbb{R}^2$ into two half-spaces $H_<$, $H_>$, and the nodes are partitioned as follows:

$$\begin{aligned}
\mathcal{V}_{bdy} &= \{V \in \mathcal{V} \cap H_< \,|\, V \text{ has a neighbor in } H_>\} \cup \{V \in \mathcal{V} \cap H_> \,|\, V \text{ has a neighbor in } H_<\},\\
\mathcal{V}_{left} &= \{V \in \mathcal{V} \cap H_<\} \setminus \mathcal{V}_{bdy}\\
\mathcal{V}_{right} &= \{V \in \mathcal{V} \cap H_>\} \setminus \mathcal{V}_{bdy}.
\end{aligned}$$

Note that due to choosing the partitioning line as the center line, we have

$$|\mathcal{V}_{left}| \approx |\mathcal{V}|/2, \qquad |\mathcal{V}_{right}| \approx |\mathcal{V}|/2, \qquad |\mathcal{V}_{bdy}| = O(\log |\mathcal{V}|). \tag{3.8}$$

(Estimates of this type are rigorously established in Lemma 3.4 below.) We then proceed as in Algorithm 3.2 by partitioning along a "center line" of the subsets of nodes (a more rigorous realization of this procedure is Algorithm 3.9 below that is based on binary space partitioning). We note that the subsets $\mathcal{V}_{left}$, $\mathcal{V}_{right}$ have a similar structure as the original set $\mathcal{V}$; thus, they are partitioned satisfying an estimate analogous to (3.8). Using this partitioning scheme in Algorithm 3.2 leads to very small frontwidths: For a mesh with $N = 2028$ nodes, a frontwidth

$\omega = 98$ is obtained (see Fig. 1 for the actual sparsity pattern). We analyze this example in more detail in Examples 4.1, 4.2 below. ■

The bounds in (3.8) are "geometrically clear." A more rigorous proof is established in the following lemma.

Lemma 3.4 *Let $\mathcal{T}$ be a geometric mesh with boundary mesh size h on a domain Ω. Fix $b \in \partial\Omega$ and choose a partitioning vector $t \neq 0$ such that the following cone condition is satisfied (cf. Fig. 2):*

$$\begin{aligned} &\exists \delta, \rho > 0, \tilde{n} \in \mathbb{R}^2 \text{ with } \langle t, \tilde{n}\rangle = 0 \text{ s.t.} \\ &\qquad C_t := \{x \in \mathbb{R}^2 \mid \langle x - b, \tilde{n}\rangle > \delta |x-b|\,|\tilde{n}|\} \cap B_\rho(b) \subset \Omega. \end{aligned} \tag{3.9}$$

Define the half-spaces

$$H_< := \{x \mid \langle x - b, t\rangle < 0\}, \qquad H_> := \{x \mid \langle x - b, t\rangle > 0\},$$

and set

$$\begin{aligned} &\mathcal{V}_\rho := \{V \mid V \text{ is a node of } \mathcal{T} \text{ and } V \in B_\rho(b)\} \\ &\mathcal{V}_{bdy} := \{V \in \mathcal{V}_\rho \cap H_< \mid V \text{ has a neighbor in } H_>\} \cup \{V \in \mathcal{V}_\rho \cap H_> \mid V \text{ has a neighbor in } H_<\}, \\ &\mathcal{V}_{left} := \{V \in \mathcal{V}_\rho \cap H_<\} \setminus \mathcal{V}_{bdy}, \qquad \mathcal{V}_{right} := \{V \in \mathcal{V}_\rho \cap H_>\} \setminus \mathcal{V}_{bdy}. \end{aligned}$$

Then

$$\begin{aligned} &|\mathcal{V}_{bdy}| \leq \gamma \log |\mathcal{V}_\rho|, \\ &\gamma^{-1}|\mathcal{V}_{left}| \leq |\mathcal{V}_{right}| \leq \gamma |\mathcal{V}_{left}| \end{aligned}$$

where $\gamma > 0$ depends only on δ, ρ, and the constants describing the geometric mesh $\mathcal{T}$. In particular, γ is independent of the point b.

Proof. Let l be the line passing through the point b with direction $\tilde{n}$, i.e., $l = \{x \in \mathbb{R}^2 \mid \langle x - b, t\rangle = 0\}$. Next, define the function

$$d(x) := \max\{h, r(x)\}, \qquad r(x) := \operatorname{dist}(x, \mathbb{R}^2 \setminus \Omega).$$

The key property of d is that

$$\forall K \in \mathcal{T}\ \forall x \in \overline{K} \qquad h_K \sim d(x). \tag{3.10}$$

Denoting by $\mathcal{K}_{bdy}$ the set of all elements that intersect the line l, $\mathcal{K}_{bdy} := \{K \in \mathcal{T} \mid \overline{K} \cap l \neq \emptyset\}$, we can bound

$$|\mathcal{V}_{bdy}| \;\leq\; 3 \sum_{K \in \mathcal{K}_{bdy}} 1 \leq C \sum_{K \in \mathcal{K}_{bdy}} \frac{1}{h_K^2} \int_K 1\, dx \leq C \sum_{K \in \mathcal{K}_{bdy}} \frac{1}{h_K^2} h_K^2 \int_K \frac{1}{d^2(x)}\, dx. \tag{3.11}$$

In order to proceed, we need two assertions:
Assertion 1: For $\delta \in (0,1)$ given by the cone condition (3.9) there holds

$$\operatorname{dist}(x, \partial\Omega) \leq \operatorname{dist}(x, b) \leq \frac{1}{\sqrt{1-\delta^2}} \operatorname{dist}(x, \partial\Omega) \qquad \forall x \in \Omega \cap l. \tag{3.12}$$

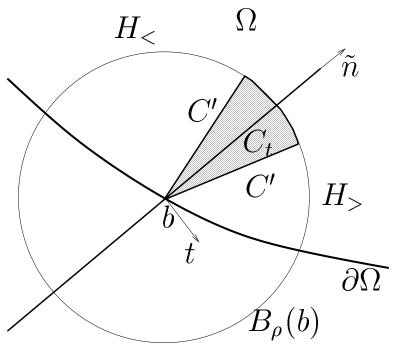

Figure 2: Notation for partitioning at a boundary point b.

The first estimate of (3.12) is obvious. For the second one, geometric considerations show for $x \in \Omega \cap l$

$$\operatorname{dist}(x,b) \leq \frac{1}{\sqrt{1-\delta^2}}\operatorname{dist}(x,C') \leq \frac{1}{\sqrt{1-\delta^2}}\operatorname{dist}(x,\partial\Omega),$$

where C' is the lateral part of ∂C_t. This proves (3.12).
Assertion 2: There exists $C > 0$ depending only on the parameters describing the geometric mesh and the parameters of the cone condition such that

$$d(x) \leq \max\{h, \operatorname{dist}(x,b)\} \leq Cd(x) \qquad \forall x \in K \quad \forall K \in \mathcal{K}_{bdy}. \tag{3.13}$$

Again, the first bound is obvious. For the second bound, let $x \in K$ for some $K \in \mathcal{K}_{bdy}$ and choose $x_K \in K \cap l$. Then

$$\operatorname{dist}(x,b) \leq h_K + \operatorname{dist}(x_K,b) \leq h_K + \frac{1}{\sqrt{1-\delta^2}}\operatorname{dist}(x_K,\partial\Omega),$$

where we used (3.12). Next, we use the properties of geometric meshes and (3.10) to get

$$\begin{aligned}\max\{h,\operatorname{dist}(x,b)\} &\leq \max\{h,h_K\} + \frac{1}{\sqrt{1-\delta^2}}\max\{h,\operatorname{dist}(x_K,\partial\Omega)\} = h_K + \frac{d(x_K)}{\sqrt{1-\delta^2}} \\ &\leq Cd(x_K) \leq Cd(x).\end{aligned}$$

Inserting this bound in (3.11) gives

$$|\mathcal{V}_{bdy}| \;\leq\; C\int_{x\in B_\rho(b)} \frac{1}{\max\{h^2,|x-b|^2\}}\,dx \leq C\int_{r=0}^{\rho}\frac{1}{\max\{h^2,r^2\}} r\,dr \leq C|\log h|.$$

Since $|\mathcal{V}_\rho| \sim h^{-1}$ (cf. [8, Prop. 2.7]), we have proved the first estimate of the lemma.
For the second estimate of the lemma, we note that the boundary parts $\Gamma_< := \partial\Omega \cap H_< \cap B_\rho(b)$ and $\Gamma_> \partial\Omega \cap H_> \cap B_\rho(b)$ have positive lengths. Thus we have $|\mathcal{V}_\rho \cap \Gamma_<| \sim h^{-1}$ and $|\mathcal{V}_\rho \cap \Gamma_>| \sim h^{-1}$ and *a fortiori* $|\mathcal{V}_{left}| \sim |\mathcal{V}_\rho| \sim |\mathcal{V}_{right}|$; the proof of the lemma is now complete. ■
The reason for the effectiveness of the partitioning strategy in Example 3.3 is that at each stage of the recursion, Algorithm 3.2 splits the set of nodes into two sets of (roughly) equal size, and a set of boundary nodes $\mathcal{V}_{bdy}$ that is very small. The property (3.8), proved in Lemma 3.4, motivates the following definition:

Definition 3.5 (**(γ, q)-balanced partitioning**) *The nested dissection Algorithm 3.2 is said to be* (γ, q)-balanced *for a set $\mathcal{V}^{(0)} = \mathcal{V} \neq \emptyset$ if at each stage i of the recursion there holds*

$$\frac{1}{\gamma}|\mathcal{V}^{(i)}_{left}| \leq |\mathcal{V}^{(i)}_{right}| \leq \gamma|\mathcal{V}^{(i)}_{left}|, \tag{3.14}$$

$$|\mathcal{V}^{(i)}_{bdy}| \leq \gamma\left[1 + \ln^q |\mathcal{V}|\right]. \tag{3.15}$$

Here, the superscripts i indicate the level of the recursion.

For a (γ, q)-balanced nested dissection algorithm, we can then show that the frontwidth grows only moderately with the problem size:

Proposition 3.6 *If the nested dissection Algorithm 3.2 is (γ, q)-balanced for $\mathcal{V}$, then the numbering generated by Algorithm 3.2 leads to a frontwidth $\omega(A)$ of the stiffness matrix A with*

$$\omega(A) \leq C_\gamma \left(1 + \ln|\mathcal{V}|\right)^{1+q}, \qquad C_\gamma := \gamma\left(1 + \frac{1}{\ln\frac{1+\gamma}{\gamma}}\right).$$

Proof. The assumption that the algorithm is (γ, q)-balanced implies easily

$$|\mathcal{V}^{(i)}_{left}| \leq \frac{\gamma}{1+\gamma}|\mathcal{V}^{(i)}|, \qquad |\mathcal{V}^{(i)}_{right}| \leq \frac{\gamma}{1+\gamma}|\mathcal{V}^{(i)}|.$$

Thus, $|\mathcal{V}^{(i)}| \leq \left(\frac{\gamma}{1+\gamma}\right)^i |\mathcal{V}|$, and the depth of the recursion is at most

$$n_{max} = \frac{1}{\ln\frac{1+\gamma}{\gamma}} \ln|\mathcal{V}|$$

since the recursion stops if $|\mathcal{V}^{(i)}| = 1$. The bound (3.7) then implies

$$\begin{aligned}\omega(A) &\leq \sum_i \gamma\left[1 + \ln^q|\mathcal{V}|\right] \leq \gamma(1 + n_{max})(1 + \ln^q|\mathcal{V}|) \\ &\leq \gamma\left(1 + \frac{1}{\ln\frac{1+\gamma}{\gamma}}\ln|\mathcal{V}|\right)(1 + \ln|\mathcal{V}|)^q \leq C_\gamma(1 + \ln|\mathcal{V}|)^{1+q},\end{aligned}$$

where we used the definition of C_γ of the statement of the proposition. ■

In Example 3.3, we studied the model situation of meshes refined toward a straight edge. In view of Lemma 3.4, we expect the partitioning strategy of be $(\gamma, 1)$-balanced. Hence, we expect the frontwidth to be of the order $O(\log^2|\mathcal{V}|)$. In Examples 4.1, 4.2, we will confirm this numerically.

3.3 Node Numbering for geometric meshes: the case $p = 1$

We now present a partitioning strategy that allows Algorithm 3.2 to be $(\gamma, 1)$-balanced for node sets that arise in the boundary concentrated FEM. The partitioning rests on the binary space partitioning (BSP), [3], which is reproduced here for convenience's sake:

Algorithm 3.7 (BSP)

```
BSP($\mathcal{X}, t$)
% input: Set of points $\mathcal{X}$, partitioning vector $t$
% output: partitioning of $\mathcal{X}$ into $\mathcal{X}_<$, $\mathcal{X}_>$, $\mathcal{X}_=$ with $|\mathcal{X}_<| \approx |\mathcal{X}_>|$ and $|\mathcal{X}_=|$ small

  1. determine the median $m$ of the set $\{\langle x, t\rangle \,|\, x \in \mathcal{X}\}$

  2. $\mathcal{X}_< := \{x \in \mathcal{X} \,|\, \langle x, t\rangle < m\}$, $\mathcal{X}_> := \{x \in \mathcal{X} \,|\, \langle x, t\rangle > m\}$, $\mathcal{X}_= := \{x \in \mathcal{X} \,|\, \langle x, t\rangle = m\}$

return
```

Remark 3.8 Since the median of a set can be determined in optimal (i.e., linear in the number of elements) complexity (see, e.g., [1, 9]), Algorithm 3.7 can be realized in optimal complexity. ■

The next algorithm formalizes our procedure of the example in Section 3.2.

Algorithm 3.9 (subdomain numbering)

```
numbering($\mathcal{V}, t, N_0$)
% input: nodes $\mathcal{V}$, vector $t$, starting number $N_0$
%output: numbering of nodes $\mathcal{V}$

if $|\mathcal{V}| = 1$ then label the element of $\mathcal{V}$ with the number $N_0$
else {
  1. $(\mathcal{X}_<, \mathcal{X}_>, \mathcal{X}_=) :=$ BSP($\mathcal{V}, t$)                % Algorithm 3.7

  2. $\mathcal{V}_{left} := \{V \in \mathcal{X}_< \,|\, \mathcal{N}(V) \subset \mathcal{X}_< \cup \mathcal{X}_=\}$,
     $\mathcal{V}_{right} := \{V \in \mathcal{X}_> \,|\, \mathcal{N}(V) \subset \mathcal{X}_> \cup \mathcal{X}_=\}$,
     $\mathcal{V}_{bdy} := \mathcal{V} \setminus (\mathcal{V}_{left} \cup \mathcal{V}_{right})$

  3. if $\mathcal{V}_{left} \neq \emptyset$ call numbering($\mathcal{V}_{left}, t, N_0$)

  4. if $\mathcal{V}_{right} \neq \emptyset$ call numbering($\mathcal{V}_{right}, t, N_0 + |\mathcal{V}_{left}|$)

  5. enumerate the elements of $\mathcal{V}_{bdy}$ starting with the number $N_0 + |\mathcal{V}_{left}| + |\mathcal{V}_{right}|$
}
return
```

Algorithm 3.9 allows us to number efficiently nodes of a mesh that is refined toward a line as in Fig. 1. Our final algorithm splits the domain into subdomains, each of which can be treated efficiently by Algorithm 3.9.

Algorithm 3.10 (node numbering)

```
  1. split the domain $\Omega$ into subdomains $\Omega_i$, $i = 1, \ldots, M$, and choose vectors $t_i \neq 0$
  2. $\mathcal{V}_i := \{V \in \mathcal{V} \cap \Omega_i \,|\, \mathcal{N}(V) \subset \overline{\Omega_i}\}$,   $i = 1, \ldots, M$
  3. $\mathcal{V}_{bdy} := \mathcal{V} \setminus \cup_{i=1}^{M} \mathcal{V}_i$
  4. $N := 1$
```

```
5. for $i = 1,\dots,M$ do {
       call numbering $(\mathcal{V}_i, t_i, N)$
       $N := N + |\mathcal{V}_i|$
   }
6. number the nodes of $\mathcal{V}_{bdy}$ starting at $N$
```

The subdomains Ω_i and the partitioning vectors t_i should be chosen such that

(a) $|\mathcal{V}_{bdy}| = O(\log|\mathcal{V}|)$;

(b) the partitioning in the subsequent calls of `numbering`$(\mathcal{V}_i, t_i, N)$ is $(\gamma, 1)$-balanced in the sense of Definition 3.5.

To obtain guidelines for the selection of subdomains Ω_i and partitioning vectors t_i, it is valuable to study examples where Condition (a) or Condition (b) are not satisfied. This is the purpose of the next example.

Example 3.11 The left and center pictures in Fig. 3 illustrate situations in which Conditions (a), (b) are violated: In the left picture of Fig. 3, the common boundary $\partial\Omega_i \cap \partial\Omega_j$ is tangential to $\partial\Omega$ at b, and thus we cannot expect $|\mathcal{V}_{bdy}| = O(\log|\mathcal{V}|)$ (cf. the cone condition (3.9)). In the center picture of Fig. 3, the partitioning vector t_i is parallel to the outer normal vector $n(x)$ at the boundary point $x \in \partial\Omega$. This prevents the partitioning from being $(\gamma, 1)$-balanced, since at some stage of the recursion, Condition (3.15) will be violated (note again the cone condition (3.9)). We refer to Example 4.5 below, where this kind of failure is demonstrated numerically. Finally, we point out that in the center picture of Fig. 3 the vector t_j for the subdomain Ω_j satisfies $|\langle t_j, n(y)\rangle| \leq \delta|t_j| < |t_j|$ for all $y \in \partial\Omega_j \cap \partial\Omega$. ∎

¿From the two cases of failure in Example 3.11, we draw the following guidelines:

1. The subdomains should be such that $\partial\Omega_i \cap \partial\Omega_j$ is non-tangential to $\partial\Omega$.
2. For each subdomain Ω_i, the partitioning vector t_i should be chosen such that the cone condition (3.9) holds uniformly in $b \in \partial\Omega_i \cap \partial\Omega$; i.e., δ, $\rho > 0$ are independent of b.

A partition chosen according to these rules is depicted in the right part of Fig. 3.

Remark 3.12 The guideline for choice t_i such that the cone condition (3.9) is satisfied at each point $b \in \partial\Omega_i \cap \partial\Omega$ guarantees that in the partitioning, $|\mathcal{V}^{(i)}_{bdy}| = O(\log|\mathcal{V}|)$ at each stage i of the recursion (see Lemma 3.4). Thus, the partitioning is $(\gamma, 1)$-balanced if we can ensure (3.14), that is, that $\mathcal{V}_{left}$ and $\mathcal{V}_{right}$ are comparable in size. Note that this could be monitored during run time. ∎

Remark 3.13 In all steps of the recursion in Algorithm 3.9, we use a fixed partitioning vector t_i. This is done for simplicity of exposition. In principle, it could be chosen differently in each step of the recursion depending on the actual set to be partitioned. Since the partitioning strategy should be $(\gamma, 1)$-balanced, one could monitor this property during run time and adjust the vector t as necessary. ∎

We conclude this section with a work estimate for the case $p = 1$:

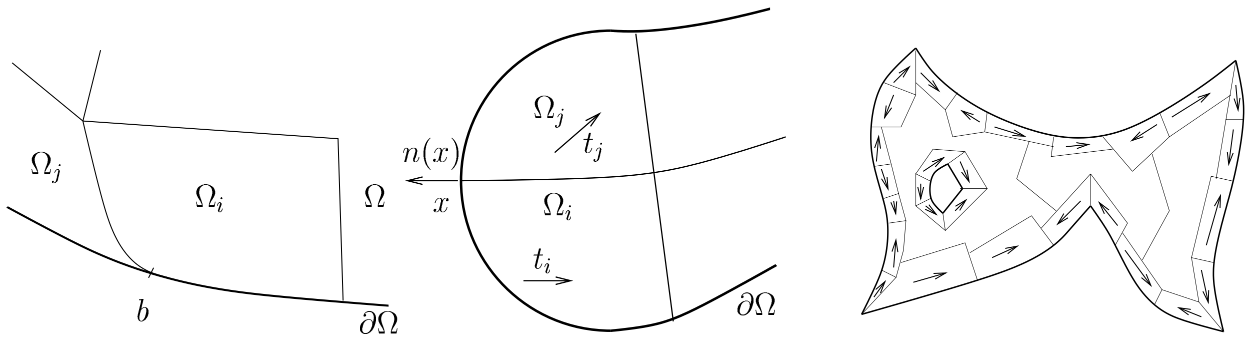

Figure 3: Choosing subdomains and partitioning vectors. Left and center: cases of failure. Right: possible partitioning with arrows indicating a good choice of vector t_i; in the three subdomains without an arrow, t_i may be chosen arbitrary.

Proposition 3.14 *Let $\mathcal{V}$ be the set of nodes corresponding to a geometric mesh on a domain Ω. Assume that the subdomains Ω_i and vectors t_i, $i = 1, \ldots, M$, in Algorithm 3.10 are chosen such that a) $|\mathcal{V}_{bdy}| = O(\log|\mathcal{V}|)$ and b) the partitioning in each call* `numbering`$(\mathcal{V}_i, t_i, N)$ *is $(\gamma, 1)$-balanced. Then the frontwidth $\omega(A)$ of the stiffness matrix on geometric meshes with $p = 1$ is bounded by*

$$\omega(A) \le C \log^2 |\mathcal{V}|,$$

where the constant C is independent of $|\mathcal{V}|$. The storage requirements nnz *and work W for the Cholesky factorization are bounded by*

$$\mathrm{nnz} \le C|\mathcal{V}| \log^2 |\mathcal{V}|, \qquad W \le |\mathcal{V}| \log^4 |\mathcal{V}|.$$

Proof. The hypothesis that Algorithm 3.10 is based on a $(\gamma, 1)$-balanced partitioning together with Proposition 3.6 implies $\omega(A) = O(\log^2 |\mathcal{V}|)$. The estimates concerning storage requirement and work then follow from Proposition 3.1. ■

Remark 3.15 On each level the nodes of $\mathcal{V}^{(i)}_{bdy}$ are numbered arbitrarily. Suitable numbering strategies of these sets could further improve the frontwidth $\omega(A)$. ■

3.4 Node Numbering: geometric meshes and linear degree vectors

We now consider the case of geometric meshes and linear degree vectors. We proceed as in Section 3.3 for the case $p = 1$ by identifying degrees of freedom with points in space. We use the notion of nodes introduced in Definition 2.7 and denote by $\mathcal{V}$ the set of all nodes. We count nodes according to their multiplicity, that is, the number of shape functions corresponding to that node. This procedure is justified by the fact that shape functions associated with the same node have the same support and therefore the same neighbors. As in the case $p = 1$, we say that node V' is the neighbor of a node V, if the intersection of the supports of the associated shape functions has positive measure. The set of neighbors of a node is defined as in (3.1).

Remark 3.16 Nodes are counted according to their multiplicity. If a one-to-one correspondence between points in space and degrees of freedom is desired, one could choose distinct nodes on an edge (e.g., uniformly distributed) to be assigned to the shape functions associated with that edge; likewise distinct nodes in an element could be selected to be assigned to shape functions associated with that element. The performance of the algorithms below will be very similar. ■

To this set of nodes and this notion of neighbors, we can apply Algorithm 3.10. In order to estimate the resulting frontwidth, we need the analog of Lemma 3.4.

Lemma 3.17 *Let $\mathcal{T}$ be a geometric mesh on a domain Ω, $\mathbf{p}$ be a linear degree vector, and assume the cone condition (3.9). Define the half-spaces*

$$H_< := \{x \,|\, \langle x-b, t\rangle < 0\}, \qquad H_> := \{x \,|\, \langle x-b, t\rangle > 0\},$$

and set

$$\begin{aligned} &\mathcal{V}_\rho := \{V \,|\, V \text{ is a node and } V \in B_\rho(v)\}, \\ &\mathcal{V}_{bdy} := \{V \in \mathcal{V}_\rho \cap H_< \,|\, V \text{ has a neighbor in } H_>\} \cup \{V \in \mathcal{V}_\rho \cap H_> \,|\, V \text{ has a neighbor in } H_<\}, \\ &\mathcal{V}_{left} := \{V \in \mathcal{V}_\rho \cap H_<\} \setminus \mathcal{V}_{bdy}, \qquad \mathcal{V}_{right} := \{V \in \mathcal{V}_\rho \cap H_>\} \setminus \mathcal{V}_{bdy}. \end{aligned}$$

Then

$$\begin{aligned} &|\mathcal{V}_{bdy}| \le \gamma \log^3 |\mathcal{V}_\rho|, \\ &\gamma^{-1}|\mathcal{V}_{left}| \le |\mathcal{V}_{right}| \le \gamma |\mathcal{V}_{left}| \end{aligned}$$

where $\gamma > 0$ depends only on δ, ρ, and the constants describing the geometric mesh $\mathcal{T}$ and the linear degree vector $\mathbf{p}$.

Proof. The proof of this lemma is very similar to that of Lemma 3.4. For the bound on $|\mathcal{V}_{bdy}|$ we have to estimate (using the notation of the proof of Lemma 3.4)

$$\sum_{K \in \mathcal{K}_{bdy}} p_K^2.$$

The desired bound then follows as in the proof of Lemma 3.4 if we observe that

$$p_K \sim 1 + \log(d(x)/h) \qquad \forall x \in K.$$

∎

In view of the appearance of the exponent 3 in Lemma 3.17, we expect Algorithm 3.10 to be $(\gamma, 3)$-balanced. In this case, we can obtain the following result for the performance of the numbering obtained by Algorithm 3.10:

Theorem 3.18 *Let $\mathcal{T}$ be a geometric mesh and $\mathbf{p}$ be a linear degree vector. Set $N = \dim S^{\mathbf{p}}(\mathcal{T}, \Omega)$. Assume that subdomains Ω_i and partitioning vectors t_i, $i = 1, \dots, M$, in Algorithm 3.10 are chosen such that a) $|\mathcal{V}_{bdy}| = O(\log^3 N)$ and b) the partitioning in each call* `numbering`$(\mathcal{V}_i, t_i, N)$ *is $(\gamma, 3)$-balanced. Then the frontwidth $\omega(A)$ of the stiffness matrix is bounded by*

$$\omega(A) \le C \log^4 N.$$

The storage requirements nnz *and work W for the Cholesky factorization are bounded by*

$$\text{nnz} \le CN \log^4 N, \qquad W \le N \log^8 N.$$

Remark 3.19 In view of Remark 3.8, Algorithm 3.10 requires $O(N \log N)$ work (i.e., optimal complexity) to compute the numbering. ∎

Remark 3.20 We assumed that the mesh consists of triangles only. However, Algorithm 3.10 can be applied to meshes containing quadrilaterals and curved elements. Theorem 3.18 holds verbatim in these cases as well. ∎

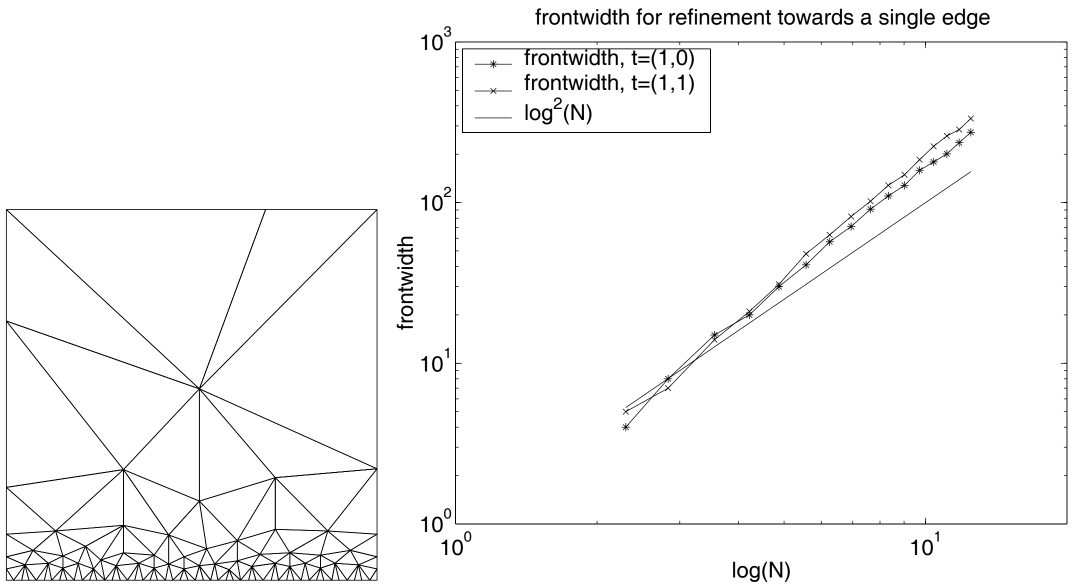

Figure 4: Examples 4.1, 4.2: influence of partitioning vector in BSP on frontwidth.

4 Numerical Examples

In this section, we confirm that the numbering obtained by Algorithm 3.10 allows for computing the Cholesky factorization with $O(N \log^q N)$, $q \in \{4, 8\}$, work. We restrict ourselves to the case $p = 1$ for simplicity; that is, we illustrate Proposition 3.14. In all examples, the nodes on the boundary correspond to unknowns, i.e., we consider Neumann problems. In all examples, we use Algorithms 3.10, 3.9 to obtain the numbering of the nodes.

In our computational experiments, the meshes are generated with the code Triangle of J.R. Shewchuck, [11]. Triangle is a realization of Ruppert's Algorithm, [10], which creates triangulations with a guaranteed minimum angle of 20°. Our reason for working with this particular triangulation algorithm is that it automatically produces meshes $\mathcal{T} = \{K\}$ with the desired property $\operatorname{diam} K \sim \operatorname{dist}(K, \partial\Omega)$ if its input consists of quasi-uniformly distributed boundary nodes only (the meshes in Figs. 5, 6, for example, are obtained with Triangle from 200 uniformly distributed points on the boundary).

In all tables and figures, N stands for the number of nodes of the mesh generated by Triangle, ω is the frontwidth of the stiffness matrix, and nnz the storage requirement for the Cholesky factor; flops is the number of multiplications, and t_{chol} the CPU-time required to perform the Cholesky factorization. We implemented the Cholesky factorization in "inner product form," where L is computed columnwise and the sparsity pattern of L is exploited.

The basic building block of our procedure is Algorithm 3.9. Our first example, therefore, analyzes in detail the situation already discussed in Example 3.3.

Example 4.1 Let $S = (0, 1)^2$ be the unit square. For $n \in \mathbb{N}$ the initial input for Triangle are the points $\{(i/n, 0) \,|\, i = 0, \ldots, n\} \cup \{(1, 0.3), (1, 1), (0.7, 1), (0, 1), (0, 0.7)\}$ (see Fig. 4 for Triangle's output for $n = 50$). The node numbering is then obtained by applying Algorithm 3.9 with the partitioning vector $t = (1, 0)^\top$. The results are collected in Table 1 and Fig. 4. In view of Proposition 3.6 we expect the frontwidth ω to be $O(\log^2 N)$. The results of Table 1 are plotted in Fig. 4, and the observed growth is indeed very close to $O(\log^2 N)$. In

		$t = (1,0)^\top$				$t = (1,1)^\top$	
n	N	ω	nnz	flops	t_{chol} [sec]	ω	nnz
4	10	4	35	$8.500e+01$	$0.000e+00$	5	41
8	17	8	95	$3.500e+02$	$0.000e+00$	7	89
16	35	15	283	$1.533e+03$	$0.000e+00$	14	293
32	68	20	759	$5.413e+03$	$0.000e+00$	21	805
64	131	30	2038	$1.993e+04$	$0.000e+00$	31	2207
128	262	41	5676	$7.363e+04$	$0.000e+00$	48	6240
256	519	57	15122	$2.600e+05$	$2.000e-02$	63	15710
512	1040	71	36792	$7.582e+05$	$4.000e-02$	82	42758
1024	2077	91	91511	$2.330e+06$	$1.300e-01$	102	107072
2048	4159	110	222924	$6.820e+06$	$4.000e-01$	128	255588
4096	8288	128	528065	$1.891e+07$	$1.130e+00$	149	598720
8192	16575	159	1259700	$5.414e+07$	$3.280e+00$	185	1471378
16384	33097	179	2843300	$1.359e+08$	$8.340e+00$	224	3486706
32768	66192	201	6624210	$3.677e+08$	$2.337e+01$	260	8030336
65536	132361	236	15264659	$9.693e+08$	$6.356e+01$	285	18334365
131072	264705	274	34604241	$2.484e+09$	$1.702e+02$	334	42163755

Table 1: Examples 4.1, 4.2: $n = \#$ points on edge, $N = \#$ mesh points, $\omega =$ frontwidth.

view of Proposition 3.1, nnz $= O(N\omega)$ and flops $\sim t_{chol} \sim W = O(N\omega^2)$; Table 1 confirms these estimates. ∎

Example 4.2 The choice in Example 4.1 of the partitioning vector $t = (1,0)^\top$ is particularly well-suited to the case of refinement toward a straight edge. In view of Lemma 3.4, we expect Algorithm 3.9 to be still $(\gamma, 1)$-balanced for partitioning vectors t that are not parallel to the normal vector. In order to illustrate that "non-optimal" choices of partitioning vectors t still lead to $(\gamma, 1)$-balanced nested dissection, we consider the same meshes as in Example 4.1 but employ Algorithm 3.9 with the vector $t = (1,1)^\top$. The results are presented in the right part of Table 1 and in Fig. 4. We observe that this choice of partitioning strategy leads to very similar results as in Example 4.1, showing robustness of our algorithm with respect to the choice of partitioning vector t. ∎

Example 4.3 In this example, the domain is the unit square $\Omega = (-0.5, 0.5)^2$. TRIANGLE's input are n uniformly distributed nodes on the boundary $\partial\Omega$ (see Fig. 5 for TRIANGLE's output for $n = 200$). The node numbering is achieved with Algorithm 3.10 for subdomains Ω_i and corresponding vectors t_i given by:

$$\begin{array}{ll} \Omega_1 := \{(x,y) \in \Omega \,|\, y < -|x|\}, & t_1 = (1,0)^\top, \\ \Omega_2 := \{(x,y) \in \Omega \,|\, x > |y|\}, & t_2 = (0,1)^\top \\ \Omega_3 := \{(x,y) \in \Omega \,|\, y > |x|\}, & t_3 = (1,0)^\top, \\ \Omega_4 := \{(x,y) \in \Omega \,|\, x < -|y|\}, & t_4 = (0,1)^\top. \end{array} \tag{4.1}$$

The numerical results are collected in Table 2. We expect the frontwidth to be $O(\log^2 N)$, which is visible in Fig. 5. ∎

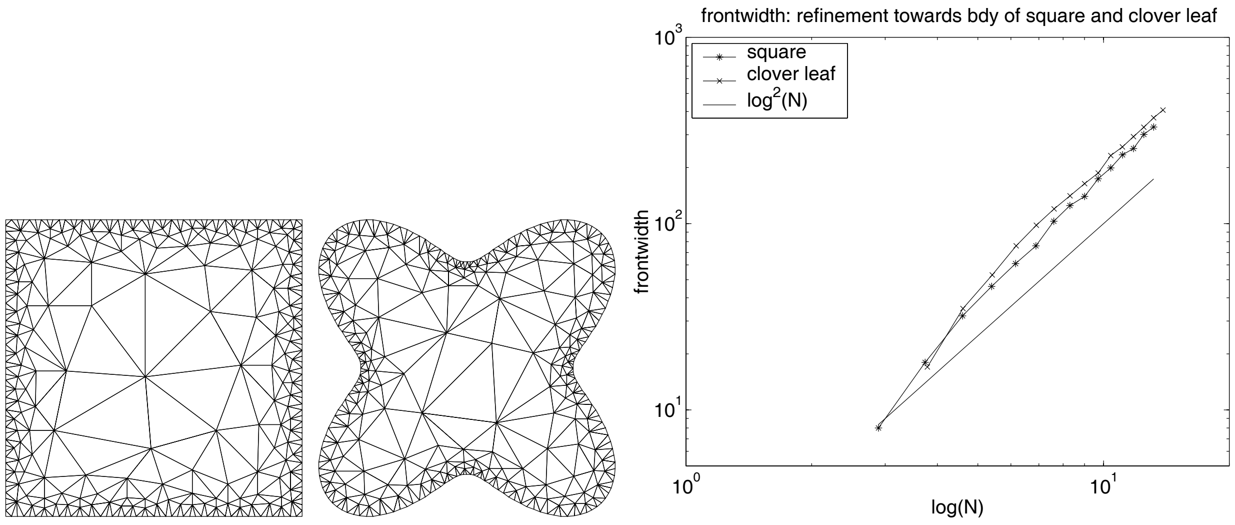

Figure 5: Examples 4.3, 4.4: meshes with 200 points on boundary (left) and frontwidth vs. $\log N$ (right).

Example 4.4 In this example, we replace the square of Example 4.3 with

$$\Omega = \{(r\cos\varphi, r\sin\varphi) \,|\, 0 \le r < 1 + 0.8\sin^2(2\varphi), \quad 0 \le \varphi < 2\pi\}.$$

(See Fig. 5 for Triangle's output for $n = 200$ boundary points.) The partitioning into four subdomains and the choice of the partitioning vectors t_i is given by (4.1). The numerical results are collected in Table 3. The expected relation $\omega = O(\log^2 N)$ is again visible in Fig. 5. ■

Example 4.5 The key feature of the choice of the partitioning vectors t_i in Examples 4.3, 4.4 is that, for each i,

$$\sup\left\{\frac{|\langle n(x), t_i\rangle|}{|t_i|} \,\middle|\, x \in \partial\Omega \cap \partial\Omega_i\right\} < 1 \qquad (n(x) \text{ is the normal at a boundary point } x). \tag{4.2}$$

This condition was identified in Example 3.11 as necessary for the binary space partitioning strategy with (fixed) vector t_i to be $(\gamma, 1)$-balanced. In this last example, we illustrate that (4.2) is indeed necessary. To that end, we replace the square of Example 4.3 with

$$\Omega_c = \{(r\cos\varphi, r\sin\varphi) \,|\, 0 \le r < 1 - c\sin^2\varphi, \quad 0 \le \varphi < 2\pi\}, \qquad c \in (0,1) \text{ fixed}.$$

The subdomains Ω_i and the partitioning vectors t_i are chosen as in Example 4.3 and given by (4.1). A calculation shows that condition (4.2) is satisfied for $c \in (0, 2/3)$ and is violated for $c = 2/3$ (see subdomains Ω_2, Ω_4 in Fig. 6). For c close to 2/3 we therefore expect our binary space partitioning to perform poorly in the sense that the sets $\mathcal{V}^{(i)}_{bdy}$ become large, thus resulting in large frontwidths. This is illustrated in Table 4, where the frontwidths for different values of c are shown in dependence on n, the number of points on the boundary. Fig. 6 shows that in the limit $c = 2/3$, the frontwidth ω does not grow polylogarithmically in N. In Table 4, we reported the number N of nodes for the case $c = 0.3$ only; we mention, however, that the meshes produced by Triangle for the three cases $c = 0.3$, $c = 0.6$, $c = 2/3$ are very similar. ■

Acknowledgments: The authors would like to thank Profs. W. Hackbusch and L.N. Trefethen for valuable comments on the paper.

n	N	ω	nnz	flops	t_{chol} [sec]
16	18	8	104	$3.960e+02$	$0.000e+00$
32	42	18	476	$3.505e+03$	$0.000e+00$
64	99	32	1803	$2.039e+04$	$0.000e+00$
128	220	46	5784	$9.254e+04$	$0.000e+00$
256	472	61	15660	$3.069e+05$	$2.000e-02$
512	979	76	40698	$9.815e+05$	$6.000e-02$
1024	1996	103	108179	$3.385e+06$	$2.000e-01$
2048	4056	125	271201	$1.028e+07$	$6.100e-01$
4096	8180	140	635551	$2.767e+07$	$1.650e+00$
8192	16476	174	1544587	$8.062e+07$	$4.960e+00$
16384	33033	199	3501495	$2.053e+08$	$1.283e+01$
32768	66085	234	8151781	$5.532e+08$	$3.683e+01$
65536	132333	253	18376760	$1.392e+09$	$9.566e+01$
131072	264580	301	41984408	$3.634e+09$	$2.582e+02$
262144	529456	330	92677554	$8.794e+09$	$6.414e+02$

Table 2: Example 4.3: $n = \#$ boundary points, $N = \#$ mesh points, $\omega =$ frontwidth.

n	N	ω	nnz	flops	t_{chol} [sec]
32	44	17	480	$3.314e+03$	$0.000e+00$
64	100	35	2003	$2.513e+04$	$0.000e+00$
128	224	53	6625	$1.214e+05$	$1.000e-02$
256	481	76	18944	$4.613e+05$	$4.000e-02$
512	994	98	49911	$1.523e+06$	$1.500e-01$
1024	2021	120	123724	$4.517e+06$	$4.700e-01$
2048	4072	141	299448	$1.278e+07$	$1.420e+00$
4096	8165	164	691256	$3.372e+07$	$3.590e+00$
8192	16417	187	1596084	$8.791e+07$	$9.380e+00$
16384	32985	232	3835565	$2.516e+08$	$2.763e+01$
32768	66062	258	8771312	$6.501e+08$	$7.367e+01$
65536	132347	293	19773070	$1.632e+09$	$1.892e+02$
131072	264614	329	44146284	$4.055e+09$	$4.804e+02$
262144	529465	370	99521860	$1.025e+10$	$1.227e+03$
524288	1059329	407	221418671	$2.512e+10$	$3.325e+03$

Table 3: Example 4.4: $n = \#$ boundary poinns[DkKWu ts, $N = \#$ mesh points, $\omega =$ frontwidth.

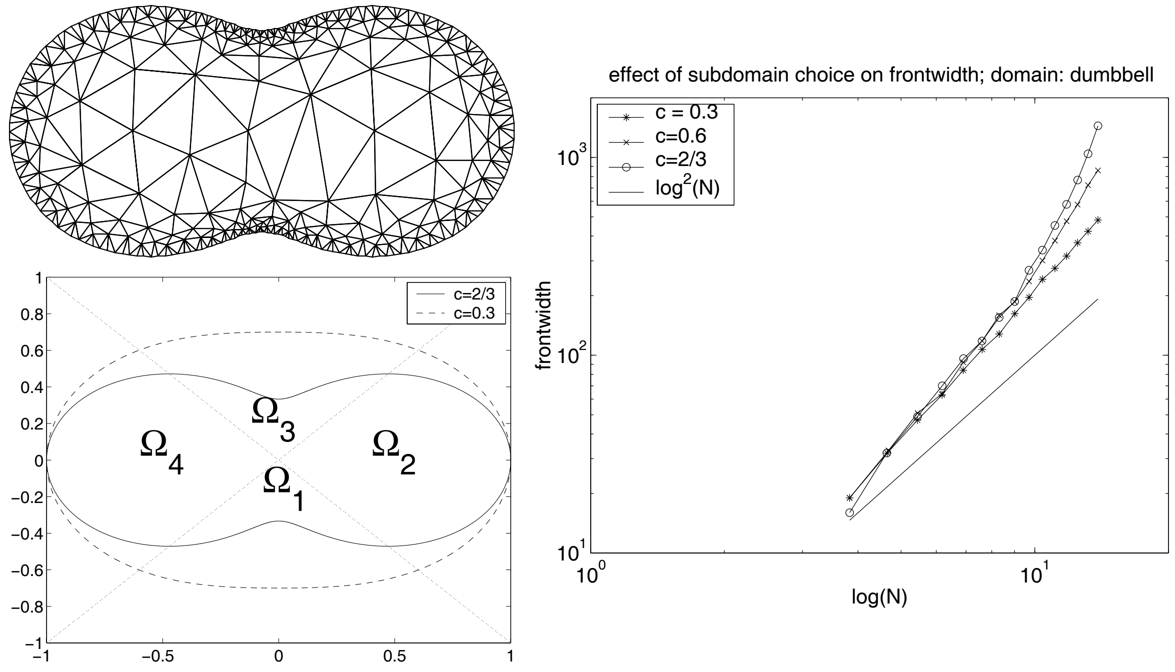

Figure 6: Example 4.5: mesh with 200 points on boundary (top left), the four subdomains for $c = 0.3$ and $c = 2/3$ (bottom left) and frontwidth vs. $\log N$ (right).

n	N	ω	ω	ω
	$c = 0.3$	$c = 0.3$	$c = 0.6$	$c = 2/3$
32	46	19	19	16
64	105	32	33	32
128	231	47	51	49
256	484	63	64	70
512	991	84	92	96
1024	2022	107	118	118
2048	4079	128	159	155
4096	8190	162	186	187
8192	16467	196	236	269
16384	32994	242	301	340
32768	66084	275	379	452
65536	132235	317	475	579
131072	264600	370	577	768
262144	529340	423	723	1043
524288	1058794	482	859	1444

Table 4: Example 4.5: $n = \#$ boundary points, $N = \#$ mesh points, $\omega =$ frontwidth.

References

[1] M. Blum, R.W. Floyd, V.R. Rivest, and R.E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, 1972.

[2] L. Demkowicz, K. Gerdes, C. Schwab, A. Bajer, and T. Walsh. A general and flexible fortran 90 *hp*-FE code. *Computing and Visualization in Science*, 1:145–163, 1998.

[3] H. Fuchs, Z.M. Kedem, and B.F. Naylor. On visible surface generation by a priori tree structures. *Computer Graphics*, 14:124–133, 1980.

[4] A. George. *Computer implementation of the finite-element method.* PhD thesis, Stanford University, 1971.

[5] A. George and J.W.H. Liu. *Computer solution of large sparse positive definite systems.* Prentice-Hall, 1981.

[6] N. E. Gibbs, W. G. Poole, and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Numer. Anal.*, 13:236–250, 1976.

[7] G.E. Karniadakis and S.J. Sherwin. *Spectral/hp Element Methods for CFD.* Oxford University Press, 1999.

[8] B.N. Khoromskij and J.M. Melenk. Boundary concentrated finite element methods. Technical Report MPI-MIS 45/2001, MPI for mathematics in the sciences, 2001.

[9] D.E. Knuth. *The Art of Computer Programming*, volume III: Sorting and Searching. Addison-Wesley, 1973.

[10] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18:548–585, 1995.

[11] J.R. Shewchuk. 2d mesh generator “triangle”. `http://www.cs.cmu.edu/~quake/triangle.html`.

[12] B. Szabó and I. Babuška. *Finite Element Analysis.* Wiley, 1991.

Max-Planck-Institute for Mathematics in the Sciences
Inselstr. 22-26, D-04103 Leipzig, Germany
`{bokh,melenk}@mis.mpg.de`