Conference Applications of Mathematics 2013
in honor of the 70th birthday of Karel Segeth.
Jan Brandts, Sergey Korotov, Michal Křížek,
Jakub Šístek, and Tomáš Vejchodský (Eds.),
Institute of Mathematics AS CR, Prague 2013

A DIRECT SOLVER FOR FINITE ELEMENT MATRICES REQUIRING $O(N \log N)$ MEMORY PLACES

Tomáš Vejchodský

Institute of Mathematics, Academy of Sciences
Žitná 25, CZ-115 61 Prague 1, Czech Republic
vejchod@math.cas.cz

Abstract

We present a method that in certain sense stores the inverse of the stiffness matrix in $O(N \log N)$ memory places, where N is the number of degrees of freedom and hence the matrix size. The setup of this storage format requires $O(N^{3/2})$ arithmetic operations. However, once the setup is done, the multiplication of the inverse matrix and a vector can be performed with $O(N \log N)$ operations. This approach applies to the first order finite element discretization of linear elliptic and parabolic problems in triangular domains, but it can be generalized to higher-order elements, variety of problems, and general domains. The method is based on a special hierarchical enumeration of vertices and on a hierarchical elimination of suitable degrees of freedom. Therefore, we call it hierarchical condensation of degrees of freedom.

1. Introduction

This paper is devoted to Prof Karel Segeth on the occasion of his 70th birthday. Karel stood at the very beginning of my scientific career as the supervisor of my Master thesis and since then we have continued to work together as collaborators and good friends until today. I am thankfull to him for many things he taught me, for a lot of help and constant support. Karel has been interested in several topics during his professional career. Efficient solution of large and sparse linear algebraic systems, which is the topic of this paper, is one of them [2, 14, 15, 18]. In addition, Karel studied the method of lines [13, 16, 19], higher-order finite elements [21], and hierarchical approaches [17, 20]. These techniques are utilized below as well.

Solvers of large and sparse linear algebraic systems stemming from discretizations of partial differential equations are considered as the bottleneck of scientific computing. Therefore, the efficiency of these solvers is of paramount importance. In this contribution we concentrate on the lowest-order triangular finite element discretization [3, 23], which is one of the most often used discretization methods that naturally yields large and sparse systems of linear algebraic equations. The sparse direct solvers and preconditioned iterative methods are two principal approaches how to solve such systems. The literature on this subject is vast. The interested reader can consult books [4, 5, 6, 10, 11, 12] and references therein.

In this contribution, we present a method that can be classified as a direct sparse solver. The idea is based on hierarchically applied static condensation of internal degrees of freedom (DOFs). The static condensation is often used in higher-order finite element methods [21], where so-called internal (or bubble) DOFs appear. These DOFs can be easily eliminated from the system in such a way that the resulting Schur complement system is of smaller dimension, better conditioned, and it keeps the original sparsity structure. See e.g. [25] for more details.

In this paper we consider the lowest-order finite element methods, where no internal DOFs exist. However, we propose to construct a hierarchy of nested meshes and consider certain DOFs of the finest mesh as internal with respect to elements of the coarser (parental) mesh. These internal DOFs can be eliminated out by the static condensation of internal DOFs. The remaining DOFs are associated with the parental mesh. Considering this mesh as the finest one, the same elimination procedure is repeated. We call this process the *hierarchical condensation of DOFs*.

During the hierarchical condensation certain auxiliary matrices are created. These matrices can be used to solve the original system with $O(N \log N)$ arithmetic operations. However, the setup of these auxiliary matrices has complexity $O(N^{3/2})$. On the other hand, they can be stored in asymptotically $O(N \log N)$ memory places. For these reasons, the hierarchical condensation of DOFs is especially useful for solving a sequence of systems with the same matrix and many different right-hand sides. For example, in the case of parabolic problems discretized in time by implicit methods.

For the sake of simplicity we present the approach using linear and symmetric parabolic problem. However, generalizations to other type of problems are possible. Generalizations to nonsymmetric, elliptic, Helmholtz, Maxwell, and similar type of problems are especially straightforward. Further, in order to simplify the description of the method, we consider triangular domains. However, generalization to arbitrary domains is not difficult. It suffices to consider an initial (coarse) mesh of the domain and apply the hierarchical condensation procedure to all triangular elements of the coarse mesh. Finally, let us note that this approach is especially advantageous in two spatial dimensions. In principal, it can be used in three and more spatial dimensions, but the resulting matrices are denser and both the memory requirements and computational complexity grow with the dimension.

The rest of this paper is organized as follows. A linear parabolic model problem is introduced in Section 2. Section 3 describes the hierarchical meshes and a special enumeration of DOFs. Section 4 forms the core of this paper and presents the hierarchical condensation of DOFs. Section 5 provides the algorithm and Section 6 computes its asymptotic complexity and memory requirements. Numerical experiments that compare the performance of various standard approaches and the hierarchical condensation of DOFs is presented in Section 7. Finally, Section 8 draws the conclusions.

2. Model problem

Let $\Omega \subset \mathbb{R}^2$ be a triangle and let $T > 0$ be fixed. We consider the following linear parabolic problem in Ω with homogeneous Dirichlet boundary conditions. Find $u = u(t, x)$ such that

$$\begin{aligned} \partial u/\partial t - \Delta u &= f \quad \text{in } (0,T) \times \Omega, \\ u(t,x) &= 0 \quad \text{for } t \in [0,T) \text{ and } x \in \partial\Omega, \\ u(0,x) &= u_0(x) \quad \text{for } x \in \Omega. \end{aligned} \tag{1}$$

In order to define the weak formulation of problem (1), we introduce the Sobolev space $V = H_0^1(\Omega)$ and assume $f \in L^2(\Omega)$ and $u_0 \in V$. The weak solution $u \in C([0,T], V)$ has the distributional time derivative $\dot{u} = \mathrm{d}u/\mathrm{d}t$ in $C([0,T], L^2(\Omega))$ and it satisfies

$$\int_\Omega \dot{u} v \,\mathrm{d}x + \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x = \int_\Omega f v \,\mathrm{d}x \quad \forall v \in V,\ t \in (0,T), \tag{2}$$

and $u(0,x) = u_0(x)$ for a.a. $x \in \Omega$.

We discretize (2) by the method of lines, see e.g. [13, 16, 19] for the works of Karel Segeth on this topic. We use the usual first-order (piecewise linear) triangular finite elements for the space discretization. Hence, we consider a triangulation $\mathcal{T}_h$ of the domain Ω and we define a subspace $V_h \subset V$ of piecewise linear functions on $\mathcal{T}_h$ by

$$V_h = \{v_h \in V : v_h|_K \in P^1(K) \text{ for all } K \in \mathcal{T}_h\},$$

where $P^1(K)$ is the three-dimensional space of linear functions in a triangle $K \in \mathcal{T}_h$. Notice that all functions $v_h \in V_h$ are continuous in Ω.

The semidiscrete solution of (2) $u_h \in C^1([0,T], V_h)$ is given by

$$\int_\Omega \dot{u}_h v_h \,\mathrm{d}x + \int_\Omega \nabla u_h \cdot \nabla v_h \,\mathrm{d}x = \int_\Omega f v_h \,\mathrm{d}x \quad \forall v_h \in V_h,\ t \in (0,T), \tag{3}$$

and $u_h(0,x) = u_{0,h}(x)$ for $x \in \Omega$, where $u_{0,h} \in V_h$ is a suitable projection of the initial condition u_0.

Equality (3) yields a system of linear ordinary differential equations. Indeed, let us define the standard finite element *hat functions* $\varphi_1, \varphi_2, \ldots, \varphi_N$ [21, 23], where $N = \dim V_h$. Each hat function $\varphi_j \in V_h$ equals to one at a vertex x_j of the mesh $\mathcal{T}_h$ and vanishes at all the other vertices. If we expand the semidiscrete solution as $u_h(t,x) = \sum_{j=1}^N y_j(t)\varphi_j(x)$ then the expansion coefficients $y = (y_1, y_2, \ldots, y_N)^T$ are determined by the system of linear differential equations

$$M\dot{y} + Ay = F, \quad y(0) = y_0, \tag{4}$$

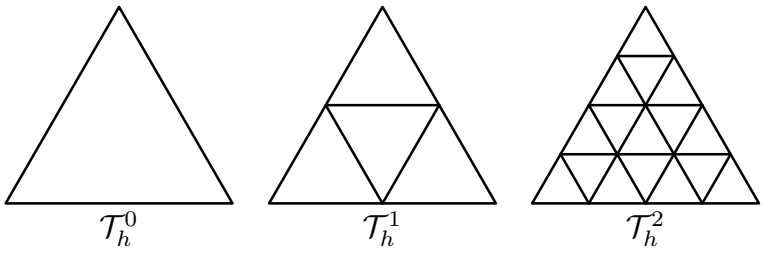

Figure 1: Triangulations of level 0, 1, and 2.

where the vector y_0 of the initial condition is determined by the expansion $u_{0,h} = \sum_{i=1}^{N} y_{0,i}\varphi_i$ of $u_{0,h}$ into the basis of V_h. Further, the mass matrix $M \in \mathbb{R}^{N\times N}$, the stiffness matrix $A \in \mathbb{R}^{N\times N}$, and the load vector $F \in \mathbb{R}^N$ have entries

$$M_{ij} = \int_\Omega \varphi_i\varphi_j \,\mathrm{d}x, \quad A_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \,\mathrm{d}x, \quad \text{and} \quad F_i = \int_\Omega f\varphi_i \,\mathrm{d}x.$$

Solving system (4) by a suitable method for systems of ordinary differential equations, we finally arrive at a fully-discrete solution to (1). In this paper we use so-called θ-method [7, 9] with a fixed time step $\tau > 0$. This method yields a system of linear algebraic equations

$$Sy^{k+1} = b^k \tag{5}$$

for the approximation y^{k+1} of y at time $t = (k+1)\tau$, $k = 0, 1, 2, \ldots$. The matrix S and the right-hand side vector b^k are given as

$$S = M + \tau\theta A, \quad b^k = \tau F + (M - \tau(1-\theta)A)y^k, \quad k = 0, 1, 2, \ldots,$$

where $\theta \in [0, 1]$ is arbitrary and fixed. Let us note that the choices $\theta = 0$, $\theta = 1/2$, and $\theta = 1$ correspond to the explicit Euler method, Crank-Nicolson method, and implicit Euler method, respectively.

In the subsequent parts of the paper we will concentrate on the hierarchical condensation of DOFs, which is an efficient method for solving the sequence of linear algebraic problems (5). Let us emphasize that we restrict ourselves to the case of simple model problem (1) for the reason of clarity only. The hierarchical condensation of DOFs can be applied to a wide class of much more general problems.

3. Mesh construction and enumeration of DOFs

The hierarchical condensation of DOFs is based on a hierarchy of successively refined and nested triangular meshes. In this section we define the triangulation, introduce its hierarchical structure represented by levels, and present a special enumeration of vertices of the triangulation (and the corresponding DOFs) that enables relatively simple implementation of the method.

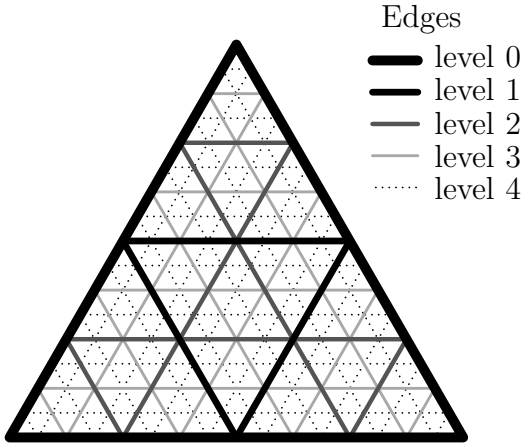

Figure 2: Levels of edges in the triangulation $\mathcal{T}_h^4$ of level 4.

The triangulation $\mathcal{T}_h$ of the triangle Ω is constructed hierarchically using *levels*. Triangulation $\mathcal{T}_h^0$ of level 0 consists of the single triangle Ω. Triangulation $\mathcal{T}_h^\ell$ of level ℓ is obtained by splitting all triangles in $\mathcal{T}_h^{\ell-1}$ into four similar subtriangles, see Figure 1. From now on we denote by $L > 0$ the fixed number of levels and we set $n = 2^L$ the number of subedges on an edge of Ω. The triangulation $\mathcal{T}_h = \mathcal{T}_h^L$ has n^2 elements, it contains $(n+2)(n+1)/2$ vertices from which $3n$ lay on the boundary $\partial\Omega$ and $(n-1)(n-2)/2$ lay in the interior of Ω. Thus, the number of DOFs is $N = \dim V_h = (n-1)(n-2)/2$, because we consider Dirichlet boundary conditions.

In order to describe the special enumeration of vertices we introduce a *level of an edge*. An edge in $\mathcal{T}_h = \mathcal{T}_h^L$ is of level $\ell = 0, 1, 2, \ldots, L$ if it lays on an edge of $\mathcal{T}_h^\ell$ but not on any edge of $\mathcal{T}_h^{\ell-1}$, $\mathcal{T}_h^{\ell-2}$, $\ldots$, $\mathcal{T}_h^0$. For example, edges of level 0 are those edges of $\mathcal{T}_h$ which lay on the boundary $\partial\Omega$. Levels of edges are indicated in Figure 2.

Level of a vertex is the smallest level of edges meeting at this vertex. Notice that any interior vertex of level $\ell = 1, 2, \ldots, L-1$ lays on two edges of level ℓ and on four edges of a higher level. Simply, all vertices of level ℓ lay on all edges of level ℓ. Since vertices correspond to the finite element basis functions and consequently to DOFs, we will naturally speak about levels of basis functions and DOFs.

The enumeration of vertices goes by levels. Since there are no vertices of level L, we first enumerate vertices of level $L-1$, then vertices of level $L-2$, etc. Finally, we enumerate vertices of level 1. Vertices of level 0 lay on the boundary of $\partial\Omega$, where we consider Dirichlet boundary conditions and hence there are no DOFs. Moreover, the enumeration of vertices of level $\ell = L-1, L-2, \ldots, 1$ goes in natural order. Precisely, there are always three interior edges of level ℓ in every element of $\mathcal{T}_h^{\ell-1}$. The enumeration of vertices of level ℓ goes by elements of $\mathcal{T}_h^{\ell-1}$. We first enumerate vertices of level ℓ on edges of level ℓ in the interior of the first element of $\mathcal{T}_h^{\ell-1}$ and then we proceed to enumerate in the same way vertices inside the second element of $\mathcal{T}_h^{\ell-1}$, etc. Figure 3 presents an example of enumeration of vertices for $L = 3$. The algorithm is as follows:

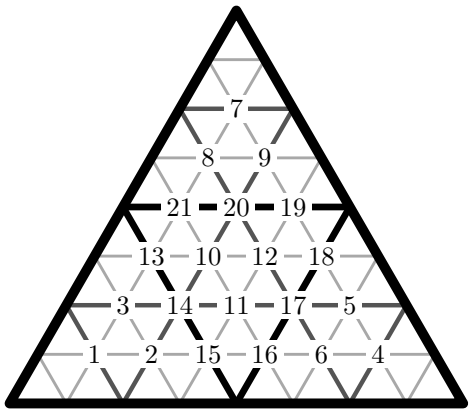

Figure 3: Triangulations $\mathcal{T}_h^3$ and enumeration of vertices. The vertices of level 2 are enumerated first (indices 1,2,...,12), then vertices of level 1 are enumerated (13,14,...,21). Notice that triplets of basis functions with indices 1,2,3; 4,5,6; 7,8,9; and 10,11,12 form bubbles in elements of triangulation $\mathcal{T}_h^1$.

for $\ell = L-1, L-2, \ldots, 1$ do
  for all elements K in $\mathcal{T}_h^{\ell-1}$ do
    enumerate all vertices lying on the three interior edges of level ℓ in K
    (there are $2^{L-\ell}-1$ vertices on each such edge and all are of level ℓ)
  end (loop through elements)
end (loop through levels)

4. Hierarchical condensation of DOFs

To describe the hierarchical condensation of DOFs, we introduce the notion of *bubble functions* or shortly *bubbles*. A function is called a bubble in element K if it is supported solely in K. The basis functions of level $L-1$ form bubbles in elements of $\mathcal{T}_h^{L-2}$. We use the static condensation to eliminate these bubbles, i.e., we eliminate all DOFs of level $L-1$. In the remaining (Schur complement) system, the DOFs of level $L-2$ correspond to bubbles in elements of $\mathcal{T}_h^{L-3}$ and they can be eliminated in the same way. This procedure continues until we traverse the whole hierarchy of meshes.

Now, we describe details of this procedure. The goal is to solve linear system

$$S_{(0)} y_{(0)} = b_{(0)}, \tag{6}$$

where $S_{(0)} = S$, $y_{(0)} = y^k$, and $b_{(0)} = b^k$ come from (5) for a fixed k. Recall that the number of DOFs (i.e. the size of this system) is $N_{(0)} = N = (n-2)(n-1)/2$. The algorithm goes in $L-2$ steps for $m = 1, 2, \ldots, L-2$.

Step 1: ($m = 1$) In this step we eliminate DOFs of level $L-1$, which form bubbles in elements of triangulation $\mathcal{T}_h^{L-2}$. The triangulation $\mathcal{T}_h^{L-2}$ consists of $(n/4)^2 = 2^{2(L-2)}$

elements and there are three vertices of level $L-1$ inside of all these elements. Thus, there are $M_{(0)} = 3 \cdot 2^{2(L-2)}$ vertices of level $L-1$. The DOFs corresponding to these vertices were enumerated first and therefore their indices are $1, 2, \ldots, M_{(0)}$. This yields the following block structure of $S_{(0)} \in \mathbb{R}^{N_{(0)} \times N_{(0)}}$, $y_{(0)} \in \mathbb{R}^{N_{(0)}}$, and $b_{(0)} \in \mathbb{R}^{N_{(0)}}$:

$$S_{(0)} = \begin{pmatrix} A_{(1)} & B_{(1)}^T \\ B_{(1)} & D_{(1)} \end{pmatrix}, \quad y_{(0)} = \begin{pmatrix} x_{(1)} \\ y_{(1)} \end{pmatrix}, \quad \text{and} \quad b_{(0)} = \begin{pmatrix} F_{(1)} \\ G_{(1)} \end{pmatrix}, \tag{7}$$

where $A_{(1)} \in \mathbb{R}^{M_{(0)} \times M_{(0)}}$, $B_{(1)} \in \mathbb{R}^{(N_{(0)} - M_{(0)}) \times M_{(0)}}$, $D_{(1)} \in \mathbb{R}^{(N_{(0)} - M_{(0)}) \times (N_{(0)} - M_{(0)})}$, $F_{(1)} \in \mathbb{R}^{M_{(0)}}$, and $G_{(1)} \in \mathbb{R}^{N_{(0)} - M_{(0)}}$. The matrix $A_{(1)}$ corresponds to bubble functions and therefore it is blockdiagonal, consisting of $(n/4)^2 = 2^{2(L-2)}$ blocks of size 3×3.

The bubble DOFs, i.e. the unknowns $x_{(1)}$, can be efficiently eliminated. The remaining DOFs, i.e. the unknowns $y_{(1)}$, are then given by a Schur complement system. To be more specific, we compute the block-wise inverse $A_{(1)}^{-1}$ and use it to obtain the Schur complement $S_{(1)}$ and the complement load $b_{(1)}$ as

$$S_{(1)} = D_{(1)} - B_{(1)} A_{(1)}^{-1} B_{(1)}^T \quad \text{and} \quad b_{(1)} = G_{(1)} - B_{(1)} A_{(1)}^{-1} F_{(1)}.$$

The two components of the coefficient vector $y_{(0)} = (x_{(1)}, y_{(1)})^T$ are then given by

$$S_{(1)} y_{(1)} = b_{(1)} \quad \text{and} \quad x_{(1)} = A_{(1)}^{-1}(F_{(1)} - B_{(1)}^T y_{(1)}).$$

Thus, as soon as the vector $y_{(1)}$ is known, the vector $x_{(1)}$ can be easily and efficiently computed. In order to compute $y_{(1)}$, we have to solve a system with matrix $S_{(1)}$. The Schur complement $S_{(1)}$ has a similar structure as the original matrix $S_{(0)}$ in the sense that vertices of level $L-2$ correspond to bubbles in the triangulation $\mathcal{T}_h^{L-3}$. Consequently, DOFs of level $L-2$ can be eliminated from $S_{(1)}$ in the same way as DOFs of level $L-1$ were eliminated from $S_{(0)}$. As a result, we obtain a Schur complement $S_{(2)}$ and the whole procedure can be repeated. In general, the m-th step of the algorithm is as follows.

Step m: (Elimination of DOFs of level $L-m$.) Put $N_{(m-1)} = N_{(m-2)} - M_{(m-2)}$ and set $M_{(m-1)} = 3(2^m - 1)(n/2^{m+1})^2 = 3(2^m - 1)2^{2(L-m-1)}$. The matrix $S_{(m-1)}$ is of size $N_{(m-1)} \times N_{(m-1)}$ and the coefficient vector $y_{(m-1)}$ and the load vector $b_{(m-1)}$ are of length $N_{(m-1)}$. There is $M_{(m-1)}$ bubble functions corresponding to vertices of level $L-m$. Thanks to the special enumeration of vertices from Section 3 the DOFs corresponding to these bubble functions have indices $1, 2, \ldots, M_{(m-1)}$ with respect to $S_{(m-1)}$. This naturally introduces the block structure

$$S_{(m-1)} = \begin{pmatrix} A_{(m)} & B_{(m)}^T \\ B_{(m)} & D_{(m)} \end{pmatrix}, \quad y_{(m-1)} = \begin{pmatrix} x_{(m)} \\ y_{(m)} \end{pmatrix}, \quad b_{(m-1)} = \begin{pmatrix} F_{(m)} \\ G_{(m)} \end{pmatrix}, \tag{8}$$

where $A_{(m)} \in \mathbb{R}^{M_{(m-1)} \times M_{(m-1)}}$, $B_{(m)} \in \mathbb{R}^{(N_{(m-1)} - M_{(m-1)}) \times M_{(m-1)}}$, etc. The matrix $A_{(m)}$ corresponds to bubble DOFs and it is block diagonal with $(n/2^{m+1})^2 = 2^{2(L-m-1)}$

blocks of size $3(2^m-1)\times 3(2^m-1)$. We invert $A_{(m)}$ and compute the Schur complement as well as the complement load as

$$S_{(m)} = D_{(m)} - B_{(m)}A_{(m)}^{-1}B_{(m)}^T \quad \text{and} \quad b_{(m)} = G_{(m)} - B_{(m)}A_{(m)}^{-1}F_{(m)}. \tag{9}$$

The components of the coefficient vector $y_{(m-1)} = (x_{(m)}, y_{(m)})^T$ are determined by

$$S_{(m)}y_{(m)} = b_{(m)} \quad \text{and} \quad x_{(m)} = A_{(m)}^{-1}(F_{(m)} - B_{(m)}^T y_{(m)}).$$

Step $L-1$: After $L-2$ steps (for $m = 1, 2, \ldots, L-2$), we are left with system

$$S_{(L-2)}y_{(L-2)} = b_{(L-2)} \tag{10}$$

with fully populated matrix $S_{(L-2)} \in \mathbb{R}^{3(n/2-1)\times 3(n/2-1)}$. We can solve this system by a standard approach such as the Cholesky decomposition for instance. As a result, we obtain the coefficients $y_{(L-2)}$ and we can compute the remaining ones by backward substitution.

Backward substitution: The remaining vectors of unknowns $x_{(m)}$, $m = L-2$, $L-3, \ldots, 1$, are easily computed as

$$x_{(m)} = A_{(m)}^{-1}(F_{(m)} - B_{(m)}^T y_{(m)}) \quad \text{and} \quad y_{(m-1)} = \begin{pmatrix} x_{(m)} \\ y_{(m)} \end{pmatrix}. \tag{11}$$

Once the matrix $S_{(0)}$ is hierarchically decomposed by the above algorithm, the next linear system with matrix $S_{(0)}$ and a different right-hand side $b_{(0)}$ can be solved very efficiently. It suffices to store matrices $Q_{(m)} = B_{(m)}A_{(m)}^{-1}$, $A_{(m)}^{-1}$, for $m = 1, 2, \ldots, L-2$, and $S_{(L-2)}^{-1}$. The given right-hand side $b_{(0)}$ is then hierarchically split into vectors $F_{(m)}$, $m = 1, 2, \ldots, L-2$, and vector $b_{(L-2)}$ using matrices $Q_{(m)}$, see (9). The final Schur complement system (10) is then solved using the stored matrix $S_{(L-2)}^{-1}$. Finally, the backward substitution (11) is performed utilizing matrices $A_{(m)}^{-1}$ and $Q_{(m)}^T$ for $m = L-2, L-1, \ldots, 1$.

Let us note that storing matrices $Q_{(m)}$ instead of $B_{(m)}$ increases the efficiency of the entire procedure significantly. On the other hand the matrix $Q_{(m)}$ has more nonzero entries than $B_{(m)}$ and its storage requires more memory. However, the difference is not large and asymptotically both these matrices have $O(N)$ nonzero entries. For details see Section 6 below.

5. Algorithm

In this section we rigorously describe the algorithm of hierarchical static condensation of DOFs with the emphasis on many linear algebraic systems with the same matrix and different right-hand side vectors. The rigorous formulation of the algorithm will be utilized in Section 6 to compute its complexity and memory requirements.

The algorithm consists of setup and solve phases. We consider the enumeration of DOFs from Section 3 and use $M_{(m-1)} = 3(2^m - 1)2^{L-m-1}$ to denote the number of bubble DOFs of level $L - m$, $m = 1, 2, \ldots, L - 2$.

First, we describe the setup phase. Its input is the matrix $S_{(0)} \in \mathbb{R}^{N_{(0)} \times N_{(0)}}$ that comes from the finite element discretization, see (6). The output consists of matrices $Q_{(m)}$, $A^{-1}_{(m)}$ for $m = 1, 2, \ldots, L-2$, and $S^{-1}_{(L-2)}$ that are needed in the solve phase.

Setup phase:

1. For $m = 1, 2, \ldots, L - 2$ do the following:
   (a) Split the matrix $S_{(m-1)}$ into blocks $A_{(m)}$, $B_{(m)}$, and $D_{(m)}$ as in (8).
   (b) Matrix $A_{(m)}$ is block-diagonal with 2^{L-m-1} blocks of size $3(2^m - 1) \times 3(2^m - 1)$. Use block-wise inversion to compute $A^{-1}_{(m)}$.
   (c) Perform the sparse matrix multiplication $Q_{(m)} = B_{(m)}A^{-1}_{(m)}$.
   (d) Compute the Schur complement matrix $S_{(m)} = D_{(m)} - Q_{(m)}B^T_{(m)}$, see (9).
   (e) Update $N_{(m)} = N_{(m-1)} - M_{(m-1)}$.
2. Compute the inverse $S^{-1}_{(L-2)}$ of the fully populated matrix $S_{(L-2)}$.
3. Output matrices $Q_{(m)}$, $A^{-1}_{(m)}$ for $m = 1, 2, \ldots, L - 2$, and $S^{-1}_{(L-2)}$.

Second, we present the solve phase. Its input data consist of a vector $b_{(0)} \in \mathbb{R}^{N_{(0)}}$, matrices $Q_{(m)}$, $A^{-1}_{(m)}$ for $m = 1, 2, \ldots, L-2$, and matrix $S^{-1}_{(L-2)}$. The output is a vector $y_{(0)}$ that solves system (6).

Solve phase:

1. For $m = 1, 2, \ldots, L - 2$ do the following:
   (a) Split vector $b_{(m-1)}$ into two blocks $F_{(m)}$ and $G_{(m)}$ as in (8).
   (b) Compute $b_{(m)} = G_{(m)} - Q_{(m)}F_{(m)}$, see (9).
   (c) Update $N_{(m)} = N_{(m-1)} - M_{(m-1)}$.
2. Solve the Schur complement problem: $y_{(L-2)} = S^{-1}_{(L-2)}b_{(L-2)}$.
3. Perform the backward substitution. For $m = L-2, L-3, \ldots, 1$ do the following:
   (a) Compute $x_{(m)} = A^{-1}_{(m)}F_{(m)} - Q^T_{(m)}y_{(m)}$.
   (b) Update $y_{(m-1)} = (x_{(m)}, y_{(m)})^T$.
4. Output vector $y_{(0)}$.

6. Computational complexity and memory requirements

In this section we compute the complexity and the memory requirements of the setup and the solve phase of the algorithm from Section 5. By the complexity we understand the asymptotic number of arithmetic operations need to perform the algorithm. The memory requirements are represented by the asymptotic number of memory places needed to store the data structures. We recall that L stands for the fixed number of levels, $n = 2^L$ denotes the number of mesh-edges on one edge of Ω, and $N = (n-2)(n-1)/2$ is the number of DOFs (the size of matrix S).

Theorem 1. *The complexity of the setup phase is $O(N^{3/2})$.*

Proof. For each $m = 1, 2, \ldots, L-2$ we invert the block diagonal matrix $A_{(m)}$. The number of arithmetic operations needed to invert a block diagonal matrix is proportional to the number of blocks multiplied by the size of each block cubed: $N_{\mathrm{op}}\left(A_{(m)}^{-1}\right) \approx 2^{2(L-m-1)} \cdot [3(2^m-1)]^3$. Further, we have to invert a dense matrix $S_{(L-2)}$. This requires $N_{\mathrm{op}}\left(S_{(L-2)}^{-1}\right) \approx [3(2^{L-1}-1)]^3$ operations. The number of arithmetic operations needed for the other steps of the setup phase is asymptotically minor with respect to $N_{\mathrm{op}}\left(A_{(m)}^{-1}\right)$ and $N_{\mathrm{op}}\left(S_{(L-2)}^{-1}\right)$. Thus, the complexity of the setup phase is

$$N_{\mathrm{op}}\left(S_{(L-2)}^{-1}\right) + \sum_{m=1}^{L-2} N_{\mathrm{op}}\left(A_{(m)}^{-1}\right) \approx (2^L)^3 = n^3 \approx N^{3/2}.$$

□

Theorem 2. *The complexity of the solve phase is $O(N \log N)$.*

Proof. The most significant operation in step 1 of the solve phase is the matrix-vector multiplication $Q_{(m)}F_{(m)}$. This multiplication requires a number of operations proportional to the number of nonzero entries in $Q_{(m)}$. It is at most twice the number of vertices of levels less then $L-m$ times the number of vertices of level $L-m$ inside one element of mesh $\mathcal{T}_h^{L-m-1}$. Thus, this number can in general reach the value up to

$$N_{\mathrm{NZ}}\left(Q_{(m)}\right) = 2 \times (N_{(m-1)} - M_{(m-1)}) \times 3(2^m - 1), \tag{12}$$

where $N_{(m-1)} = 3(2^m-1)2^{L-m-1}(2^{L-m}-1)$ is the number of vertices of levels less than or equal to $L-m$ and $M_{(m-1)} = 3(2^m-1)2^{2(L-m-1)}$ is the number of vertices of level $L-m$. Consequently, $N_{(m-1)} - M_{(m-1)} = 3(2^m-1)(2^{2(L-m-1)} - 2^{L-m-1})$. Since the matrix-vector multiplication $Q_{(m)}F_{(m)}$ is performed for $m = 1, 2, \ldots, L-2$, the complexity of step 1 is proportional to

$$\sum_{m=1}^{L-2} N_{\mathrm{NZ}}\left(Q_{(m)}\right) = \sum_{m=1}^{L-2} 18(2^m-1)^2\left(2^{2(L-m-1)} - 2^{L-m-1}\right) = (9L-42)2^{2L-1}$$

$$+ (18L+9)2^L + 12 = \frac{n^2}{2}(9\log_2 n - 42) + n(18\log_2 n + 9) + 12 \approx N \log N. \tag{13}$$

In step 2 we multiply the vector $b_{(L-2)}$ by the fully populated matrix $S^{-1}_{(L-2)}$ of size $3(2^{L-1}-1)\times 3(2^{L-1}-1)$. The complexity of this operation is

$$3^2(2^{L-1}-1)^2 \approx n^2 \approx N.$$

In step 3 we perform matrix-vector multiplications with matrices $A^{-1}_{(m)}$ and $Q^T_{(m)}$ for $m = L-2, L-3, \ldots, 1$. The complexity of multiplication by matrix $A^{-1}_{(m)}$ is proportional to the number of its nonzero entries, which is less than the number of nonzero entries of $Q^T_{(m)}$. Multiplications by the matrices $Q^T_{(m)}$ and $Q_{(m)}$ are of the same complexity proportional to $N_{\mathrm{NZ}}\left(Q_{(m)}\right)$, see (12). Thus, the complexity of step 3 is proportional to $N\log N$ as in step 1. Consequently, the total complexity of the solve phase is $O(N\log N)$. □

Theorem 3. *The memory requirements to store matrices $Q_{(m)}$, $A^{-1}_{(m)}$ for $m = 1, 2, \ldots, L-2$, and matrix $S^{-1}_{(L-2)}$ are $O(N\log N)$.*

Proof. The fully populated matrix $S^{-1}_{(L-2)}$ contains $N_{\mathrm{NZ}}\big(S^{-1}_{(L-2)}\big) = (3(n/2-1))^2 = 9/4n^2 - 9n + 9$ entries. The number of nonzero entries in matrix $A^{-1}_{(m)}$ is equal to the number of its blocks times the size of the block squared, i.e. $N_{\mathrm{NZ}}\big(A^{-1}_{(m)}\big) = 2^{2(L-m-1)}[3(2^m-1)]^2$. Thus, for all $m = 1, 2, \ldots, L-2$ we have

$$N_{\mathrm{NZ}}\left(A^{-1}\right) = \sum_{m=1}^{L-2} N_{\mathrm{NZ}}\big(A^{-1}_{(m)}\big) = 2^{2L-2}(9L-33) + 18\cdot 2^L - 12$$
$$= \frac{n^2}{4}(9\log_2 n - 33) + 18n - 12.$$

The total number of nonzero entries in all matrices $Q_{(m)}$ for $m = 1, 2, \ldots, L-2$ was computed above, see (13). Hence, we can conclude that the total amount of memory places needed to store matrices $Q_{(m)}$, $A^{-1}_{(m)}$ for $m = 1, 2, \ldots, L-2$, and matrix $S^{-1}_{(L-2)}$ is asymptotically proportional to $N\log N$. □

Let us note that the original stiffness matrix $S_{(0)}$ contains $N_{\mathrm{NZ}}\left(S_{(0)}\right) = 7(n-2)(n-1)/2 - 6(n-2) = (7n^2-33n+38)/2$ nonzero entries. Making rough estimates and considering a sufficiently high number of levels L, we may say that the total memory requirements to store matrices $Q_{(m)}$, $A^{-1}_{(m)}$ for $m = 1, 2, \ldots, L-2$, and $S^{-1}_{(L-2)}$ are about $2(L-4)$ times higher than $N_{\mathrm{NZ}}\left(S_{(0)}\right)$.

7. Numerical experiments

In this section we compare the performance of the above described hierarchical condensation of DOFs with standard methods. The numerical tests are done in Matlab and the hierarchical condensation is compared with standard Matlab implementations of fully populated matrix inversion, sparse Cholesky factorization,

conjugate gradients (CG) preconditioned by the incomplete Cholesky factorization and an optimized direct sparse solver (backslash command in Matlab).

We consider the parabolic problem (1) in a triangle Ω with vertices $[0,0]$, $[1,0]$, and $[0.7,0.8]$ for $t \in (0,100)$. The right-hand side and the initial conditions are chosen as $f = 1$ and $u_0 = \lambda_1^4 \lambda_2 \lambda_3$, where $\lambda_1, \lambda_2, \lambda_3$ are barycentric coordinates in Ω. This problem is discretized as described in Section 2. We use the time step $\tau = 0.1$ and the Crank-Nicolson method ($\theta = 1/2$) for time discretization. The space discretization is done on a sequence of uniform and successively refined meshes. We construct the meshes as described in Section 3 for the number of levels $L = 4, 5, \ldots, 10$.

During the time evolution, it is necessary to solve system (5) in total 1000 times (the final time is $T = 100$ and the time step is $\tau = 0.1$). Before the first system (5) is solved, we perform a setup phase for the given matrix S, store the necessary data, and then we run the solve phase 1000 times.

The first of the tested methods is to compute the fully populated matrix inverse in the setup phase, store the inverse, and then just multiply the right-hand side by this inverse in the solve phases. This method is very inappropriate for the presented problem, because it ignores the sparsity of matrix S. We include it in the test in order to illustrate the magnitude of this inappropriateness.

The second method is a sparse Cholesky factorization with the approximate minimum degree permutation trying to minimize the fill-in. A suitable permutation and the Cholesky factor of permuted S are computed in the setup phase. The stored permutation and the Cholesky factor are then used in the solve phases.

The third method is the CG method preconditioned by incomplete Cholesky factorization with no fill-in. In the setup phase we construct the preconditioner, store it, and use it in the solve phases. The initial approximation is taken from the previous time step and the CG iterations are stopped as soon as the relative residual decreases below 10^{-6}. This was always happening in a few iterations.

The fourth method is the backlash command of Matlab. It is a highly optimized procedure combining various sparse direct solvers for various types and sizes of matrices. We perform no setup phase and use the backslash command directly in the solve phases. Finally, the fifth method is the hierarchical condensation of DOFs as described above.

Figure 4 presents the CPU-times required to perform the setup phase (left panel) and the CPU-times for 1000 solve phases (right panel). We see that the hierarchical condensation has the fastest solve phase of all tested methods. It is more than two times faster than the sparse Cholesky factorization. The setup phase of the hierarchical condensation is the second fastest after the sparse Cholesky factorization. However, the asymptotic complexity of the setup phases of the hierarchical condensation and sparse Cholesky factorization seems to be the same in this example. The other methods are not competitive with the exception of the preconditioned CG. Its performance during the solve phase is relatively improving with growing number of DOFs, however the complexity of its setup phase is considerably higher than the complexity of the setup phase of both sparse Cholesky factorization and hierarchi-

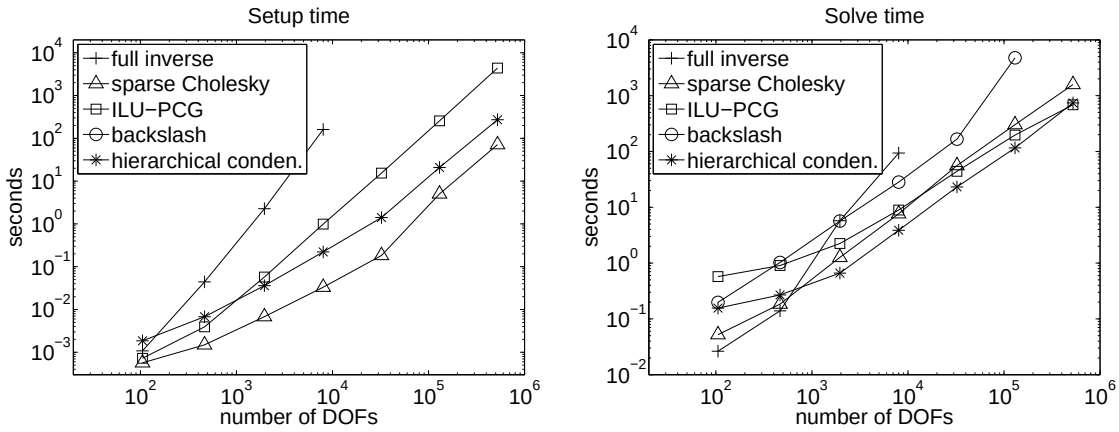

Figure 4: CPU-times for the setup phase (left) and for the 1000 of solve phases (right). Notice the logarithmic scales and zero setup time for the backslash solver.

cal condensation. Let us note that the full-inverse approach runs over the available memory starting from the number of levels $L = 8$.

Further, let us note that no special effort was made to optimize the Matlab code of hierarchical condensation for speed. The code is relatively simple, it contains a few short for-cycles and the majority of CPU-time is spent by various sparse matrix operations. Therefore, we assume that the differences due to the compiled codes (inversion, Cholesky factorization, and backlash) and interpreted codes (preconditioned CG and hierarchical condensation) are not fundamental.

8. Conclusions

In this paper we presented a hierarchical condensation of DOFs, which is a direct sparse method for solving linear algebraic systems. We prove that the setup phase requires $O(N^{3/2})$ arithmetic operations, the resulting data are stored in $O(N \log N)$ memory places, and the solve phase takes $O(N \log N)$ operations, where N stands for the number of DOFs.

The method was presented using a simple model problem, a triangular domain, and the lowest-order finite element method. However, generalizations to more general problems and domains are straightforward as well as generalization to higher-order finite elements. Generalizations to higher spatial dimensions are possible as well.

A clear bottleneck of this approach is the setup phase which has a suboptimal complexity, because the expected optimal complexity would be $O(N)$, see e.g. multigrid approaches [1, 24] or optimal methods in special domains [8, 22].

Nevertheless, the hierarchical condensation of DOFs provides an insight into the structure of the inverse of the finite element matrices. We believe that this insight can be fruitful if it enables to modify the setup phase such that it is performed approximately and fast. This approximate inverse can then serve as an efficient and hopefully optimal preconditioner.

References

[1] Briggs, W. L., Henson, V. E., and McCormick, S. F.: *A multigrid tutorial*, 2nd ed. SIAM, Philadelphia, PA, 2000.

[2] Červ, V. and Segeth, K.: A comparison of the accuracy of the finite-difference solution to boundary value problems for the Helmholtz equation obtained by direct and iterative methods. Apl. Mat. **27** (1982), 375–390.

[3] Ciarlet, P. G.: *The finite element method for elliptic problems.* North-Holland Publishing Co., Amsterdam, 1978.

[4] Duff, I., Erisman, A., and Reid, J.: *Direct methods for sparse matrices.* Clarendon Press, Oxford, 1986.

[5] Greenbaum, A.: *Iterative methods for solving linear systems.* SIAM, Philadelphia, PA, 1997.

[6] Hackbusch, W.: *Iterative solution of large sparse systems of equations. Transl. from the German.* Springer-Verlag, New York, NY, 1994.

[7] Hairer, E. and Wanner, G.: *Solving ordinary differential equations. II: Stiff and differential-algebraic problems. 2nd rev. ed.* Springer, Berlin, 1996.

[8] Hockney, R.: A fast direct solution of Poisson's equation using Fourier analysis. J. Assoc. Comput. Mach. **12** (1965), 95–113.

[9] Lambert, J.: *Numerical methods for ordinary differential systems: the initial value problem.* John Wiley & Sons, Chichester, 1991.

[10] Meurant, G.: *Computer solution of large linear systems.* Elsevier, Amsterdam, 1999.

[11] Osterby, O. and Zlatev, Z.: *Direct methods for sparse matrices.* Springer-Verlag, Berlin, 1983.

[12] Saad, Y.: *Iterative methods for sparse linear systems*, 2nd ed. SIAM, Philadelphia, PA, 2003.

[13] Segeth, K.: A posteriori error estimation with the finite element method of lines for a nonlinear parabolic equation in one space dimension. Numer. Math. **83** (1999), 455–475.

[14] Segeth, K.: On the choice of iteration parameters in the Stone incomplete factorization. Apl. Mat. **28** (1983), 295–306.

[15] Segeth, K.: Numerical experiments with the Stone incomplete triangular decomposition. In: *Mathematical models in physics and chemistry and numerical methods of their realization (Visegrád, 1982)*, *Teubner-Texte Math.*, vol. 61, pp. 226–236. Teubner, Leipzig, 1984.

[16] Segeth, K.: A posteriori error estimates for parabolic differential systems solved by the finite element method of lines. Appl. Math. **39** (1994), 415–443.

[17] Šolín, P. and Segeth, K.: A new sequence of hierarchic prismatic elements satisfying de Rham diagram on hybrid meshes. J. Numer. Math. **13** (2005), 295–317.

[18] Šolín, P. and Segeth, K.: Performance of various ODE solvers on FV-semidiscretized nonstationary compressible Euler equations. Acta Tech. CSAV **47** (2002), 47–66.

[19] Šolín, P. and Segeth, K.: Application of the method of lines to unsteady compressible Euler equations. Internat. J. Numer. Methods Fluids **41** (2003), 519–535.

[20] Šolín, P. and Segeth, K.: Hierarchic higher-order Hermite elements on hybrid triangular/quadrilateral meshes. Math. Comput. Simulation **76** (2007), 198–204.

[21] Šolín, P., Segeth, K., and Doležel, I.: *Higher-order finite element methods*. Studies in Advanced Mathematics, Chapman & Hall/CRC, Boca Raton, FL, 2004.

[22] Swarztrauber, P. N.: The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. SIAM Rev. **19** (1977), 490–501.

[23] Szabó, B. and Babuška, I.: *Finite element analysis*. A Wiley-Interscience Publication, John Wiley & Sons Inc., New York, 1991.

[24] Trottenberg, U., Oosterlee, C. W., and Schüller, A.: *Multigrid. with guest contributions by A. Brandt, P. Oswald, K. Stüben.* Academic Press, Orlando, FL, 2001.

[25] Vejchodský, T. and Šolín, P.: Static condensation, partial orthogonalization of basis functions, and ILU preconditioning in the *hp*-FEM. J. Comput. Appl. Math. **218** (2008), 192–200.