# A Domain Decomposition Approach for Calculating the Graph Corresponding to a Fibrous Geometry

Achi Brandt[1], Oleg Iliev[2], and Joerg Willems[3]

[1] Department of Applied Mathematics & Computer Science, The Weizmann Institute of Science, 76100 Rehovot, Israel achi.brand@tweizmann.ac.il
[2] Fraunhofer Institut für Techno- und Wirtschaftsmathematik, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany oleg.iliev@itwm.fraunhofer.de
[3] Fraunhofer Institut für Techno- und Wirtschaftsmathematik, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany joerg.willems@itwm.fraunhofer.de

## 1 Introduction

The effective properties of composite materials/media are in strong demand in engineering, geoscience, and environmental studies to name just a few examples. In [2], we presented an efficient algorithm for computing an approximation of the effective thermal conductivity tensor for high contrast fibrous geometries. The essential idea of the approach is to take into consideration the network-like structure of a given fibrous geometry and to perform all calculations on the induced unstructured grid. More precisely, the intersections of fibers are considered as nodes and the connecting fibers between nodes are considered as edges of an undirected graph. The weight of each edge depends on the diameter and the conductivity of the respective fiber and the distance of the connected nodes. A comparison between the results produced by our algorithm and classical methods, which resolve the fibrous geometry using volumetric elements, yields evidence of its efficiency and reliability for a large class of problems from engineering and science.

In the article at hand the primary focus is on increasing the computational efficiency of the essential preprocessing step, i.e., of setting up the graph. In [2] the computation of the fiber intersections is carried out straightforwardly, i.e., each fiber is compared against any other fiber for intersection. This preprocessing stage, if carried out like this, has a complexity which is quadratic in the number of fibers and can therefore, for samples with very many fibers, become prohibitively expensive.

The idea to reduce the complexity of the straightforward strategy discussed above is to partition the domain into a grid of coarse cells. Then by going along each fiber, we determine the coarse grid cells through which this fiber passes. Once this has been completed we go through each coarse cell and check for intersections only among those fibers passing through one and the same cell. This is done in such a way that two fibers are compared only once, no matter if they mutually lie in several coarse cells. The resulting graph is - except for the ordering of the nodes - identical

to the one computed by the standard approach. The computational cost, however, is significantly reduced.

The remainder of the article at hand is organized as follows: In section 2 we introduce some notations and definitions needed for presenting our argument. After that we give a short description of the problem which we are ultimately interested in solving. In section 4 the algorithm that we use to construct the graphs corresponding to fibrous geometries is discussed. In a subsection we also provide a short analysis of the computational cost of this algorithm. The final section of this paper is devoted to numerical results and conclusions.

## 2 Preliminaries

For the arguments to follow we would like to introduce some notations and definitions. In order to make the presentation somewhat simpler we restrict our exposition to three spatial dimensions, which is anyway the most interesting case from a practical point of view. In [2] random fibrous geometries and the computation of their effective thermal conductivities are the main targets of consideration. Let us now briefly discuss what exactly we mean by a fiber and a random fibrous geometry.

By a fiber $\varphi$, we mean a cylindrical object of finite or infinite length. In particular, a fiber is supposed to have a straight line at its center. To generate a fibrous geometry these objects are randomly "thrown into" our domain $\Omega$ and cut-off at the boundary $\partial\Omega$. For simplicity we assume $\Omega$ to be brick shaped. The collection of all fibers in $\Omega$ is denoted by $\Phi$. Let the set of all intersections of the straight lines at the centers of fibers with $\partial\Omega$ be denoted by $\partial\omega$. The actual numerical generation of our fibrous geometries is done by the GeoDict2008 software (For more information see http://www.geodict.com.) With this random construction different fibers may and in general will intersect.

Now, let $\omega$ be the set of points, where two or more fibers cross. For a simpler presentation and to avoid unnecessary technicalities, we assume, that whenever two fibers (i.e., the cylindrical objects) have a nonempty intersection the same holds true for their center lines. For a randomly generated fibrous geometry this assumption will in general not be satisfied. In practice, however, this doesn't pose any serious difficulties. In order to determine whether two fibers cross, we calculate the distance between their center lines. If this distance is smaller than the sum of the fiber radii, we say that the fibers cross and for each of the involved center lines we store the point at which they are closest, i.e., the distance of these points is equal to the distance of the center lines of the involved fibers. The crossing node is then set to be in the middle of these two points. We also define $\overline{\omega} := \omega \cup \partial\omega$ to be the set of all internal and boundary intersections.

Let $h$ be the characteristic distance between adjacent (i.e., adjacent on a fiber) nodes in $\overline{\omega}$ and let $d$ be the characteristic diameter of all fibers in $\overline{\Omega}$. We require $d \ll h$ in order to have a meaningful notion of a graph induced by the fibers (which correspond to the edges of the graph) and their intersections (which correspond to the nodes of the graph).

## 3 Statement of the Problem

It is a well-known result from homogenization theory (cf. e.g. [3] and the references therein) that the effective conductivity tensor $\widetilde{K}$ for a periodic or statistically homogeneous medium can be calculated by

$$\widetilde{K}\mathbf{e}_i = \langle K\nabla u_i \rangle_{\Omega}, \quad i = 1,2,3,$$

where $\mathbf{e}_i$ is the $i$-th unit vector, $K$ denotes the fine scale conductivity, $\langle\cdot\rangle_{\Omega}$ is the volumetric average over $\Omega$, and $u_i$ solves

$$\begin{aligned} \nabla\cdot(K\nabla u_i) &= 0 \quad \text{in } \Omega \\ u_i(\mathbf{x}) &= x_i \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

with $x_i$ being the $i$-th component of $\mathbf{x}$.

In [1] it was shown that for a composite medium having a (very) high contrast in conductivities, the effective conductivity tensor of the entire medium can be approximated by solving three (one for each spatial dimension) constant coefficient elliptic problems. These constant coefficient problems are posed only on the highly conductive parts of $\Omega$. Several numerical examples in [1] show that this approach yields very good results for a class of problems interesting from a scientific and engineering point of view.

Departing from the framework in [1] an algorithm was developed in [2] specifically designed for approximating the effective conductivity tensors for high contrast fibrous geometries. The essential idea is to perform all calculations on the graph induced by the underlying fibrous structure. The discrete problems corresponding to (1) read as follows:

$$\begin{aligned} \mathcal{D}(K\mathcal{G}y_i) &= 0 \quad \text{in } \omega \\ y_i &= x_i \quad \text{on } \partial\omega, \end{aligned} \tag{2}$$

where $\mathcal{D}$ and $\mathcal{G}$ are discrete versions of the divergence and gradient operator, respectively, having values on the nodes (●) and faces (|) between adjacent nodes, respectively (see Fig. 1). For a precise definition of $\mathcal{D}$ and $\mathcal{G}$ as well as for an error analysis we would like to refer the reader to [2].

## 4 A Divide and Conquer Algorithm

The computational bottleneck of the algorithm discussed in [2] is the preprocessing step of setting up the graph, i.e., the computation of the set of intersections $\omega$. If this is done in a straightforward way, meaning by comparing each fiber with every other, the computational cost is $\mathcal{O}(n_\Phi^2)$, where $n_\Phi$ is the number of fibers in $\Omega$. For large geometries with very many fibers this will of course soon become prohibitively expensive.

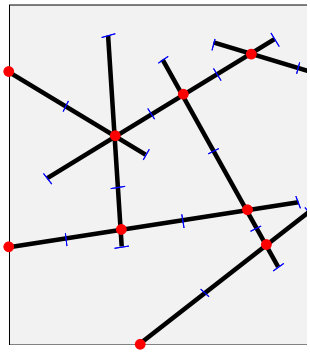


**Fig. 1.** Fibrous structure with induced nodes and faces.

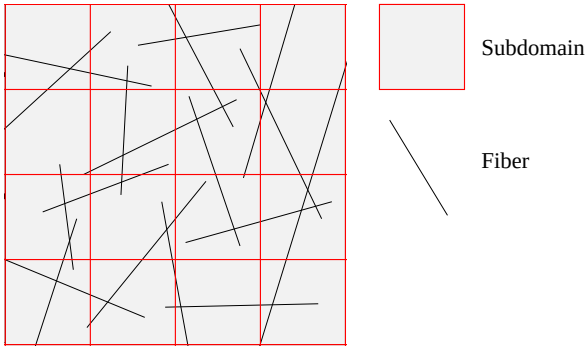



**Fig. 2.** Subdomains $\Omega_{\mathbf{j}}$ and fibers $\varphi_i$.

The idea to cure this problem is to divide our domain $\Omega$ into subdomains $\Omega_{\mathbf{j}}$, $\mathbf{j} \in \{1,\dots n_{\Omega,x_1}\} \times \{1,\dots,n_{\Omega,x_2}\} \times \{1,\dots,n_{\Omega,x_3}\} =: \mathscr{J}$, where $n_{\Omega,x_1}$, $n_{\Omega,x_2}$, and $n_{\Omega,x_3}$ are the number of subdomains in each spatial direction, $\mathbf{j}$ is a multi-index, and $\cup_{\mathbf{j}\in\mathscr{J}} \Omega_{\mathbf{j}} = \Omega$ (cf. Fig. 4). For simplicity, we again suppose that $\Omega_{\mathbf{j}}$ is brick shaped. Then for each fiber we check through which subdomains it passes and construct the sets $\Phi_{\mathbf{j}}$, where $\Phi_{\mathbf{j}}$ denotes the set of fibers passing through $\Omega_{\mathbf{j}}$. Then for each $\mathbf{j} \in \mathscr{J}$ we check for intersections among all $\varphi \in \Phi_{\mathbf{j}}$. In Algorithm 1 we make these considerations more formal.

*Remark 1.* The condition $\lambda_2(\Omega_{\mathbf{j}} \cap \Omega_{\tilde{\mathbf{j}}}) \neq 0$ in step 8 of Algorithm 1 means that we only check adjacent subdomains which have a common face with the previous subdomain. We don't need to take into consideration those adjacent subdomains which only have a common edge or point. This is because fibers are volumetric objects. In particular they have a strictly positive diameter.

*Remark 2.* It should be noted that the standard straightforward approach of testing each fiber with any other for intersection is a special case of Algorithm 1 – consider the case $\#\mathscr{J} = 1$.

### 4.1 Numerical Complexity of Algorithm 1

Now, we would like to obtain an estimate of the numerical cost of Algorithm 1 in order to be able to compare it with the complexity of the straightforward approach of checking each fiber with respect to any other one for intersection. It is evident, that this straightforward approach requires $\mathcal{O}(n_\Phi^2)$ operations.

Since for general randomly generated fiber geometries the computation of the numerical complexity of Algorithm 1 would go into too much detail concerning the generation of such geometries, we perform our analysis only for one particular structure with regularly arranged infinitely long fibers (cf. Fig. 5). More pre-

1: $\Phi_{\mathbf{j}} = \emptyset \; \forall \mathbf{j} \in \mathscr{J}$
2: **for** $i = 1, \ldots, n_\Phi$ **do**
3: Compute an end point $\mathbf{x}_i$ of $\varphi_i$ and determine $\mathbf{j} \in \mathscr{J}$ such that $\mathbf{x}_i \in \Omega_{\mathbf{j}}$.
4: Set $\Phi_{\mathbf{j}} = \Phi_{\mathbf{j}} \cup \{\varphi_i\}$, i.e., add $\varphi_i$ to the set of fibers passing through $\Omega_{\mathbf{j}}$.
5: Set $\tilde{\mathscr{J}} = \{\mathbf{j}\}$. The subdomains corresponding to $\tilde{\mathscr{J}}$ are those intersected by $\varphi_i$ and having at least one neighbor which hasn't been checked for intersection with $\varphi_i$, yet.
6: **while** $\# \tilde{\mathscr{J}} \neq 0$ **do**
7: **for** $\mathbf{j} \in \tilde{\mathscr{J}}$ **do**
8: Let $\hat{\mathscr{J}}$ be the set of all $\hat{\mathbf{j}}$ such that $\lambda_2(\Omega_{\mathbf{j}} \cap \Omega_{\hat{\mathbf{j}}}) \neq 0$ and $\varphi_i \notin \Phi_{\hat{\mathbf{j}}}$, where $\lambda_2$ is the two-dimensional Lebesgue measure. The subdomains corresponding to $\hat{\mathscr{J}}$ are those neighbors of $\Omega_{\mathbf{j}}$ for which intersection with $\varphi_i$ hasn't been verified yet.
9: **for** $\hat{\mathbf{j}} \in \hat{\mathscr{J}}$ **do**
10: **if** $\varphi_i$ crosses $\Omega_{\hat{\mathbf{j}}}$ **then**
11: Set $\Phi_{\hat{\mathbf{j}}} = \Phi_{\hat{\mathbf{j}}} \cup \{\varphi_i\}$, i.e., $\varphi_i$ is added to the set of fibers passing through $\Omega_{\hat{\mathbf{j}}}$.
12: Set $\tilde{\mathscr{J}} = \tilde{\mathscr{J}} \cup \{\hat{\mathbf{j}}\}$. Since $\Omega_{\hat{\mathbf{j}}}$ is intersected by $\varphi_i$ we now in turn need to check the neighbors of $\Omega_{\hat{\mathbf{j}}}$ for intersection with $\varphi_i$, too.
13: **end if**
14: **end for**
15: Set $\tilde{\mathscr{J}} = \tilde{\mathscr{J}} \backslash \{\mathbf{j}\}$. Since all neighbors of $\Omega_{\mathbf{j}}$ have been checked for intersection with $\varphi_i$, $\mathbf{j}$ is removed from $\tilde{\mathscr{J}}$.
16: **end for**
17: **end while**
18: **end for**
19: **for** $\mathbf{j} \in \mathscr{J}$ **do**
20: **for** $\varphi_i \in \Phi_{\mathbf{j}}$ **do**
21: **for** $\varphi_k \in \Phi_{\mathbf{j}}$ and $k > i$ **do**
22: **if** $\varphi_k$ and $\varphi_i$ haven't been tested for intersecting yet **then**
23: Test $\varphi_k$ and $\varphi_i$ for intersection and add a corresponding node to the graph if the fibers cross.
24: **end if**
25: **end for**
26: **end for**
27: **end for**

Algorithm 1: Compute a graph corresponding to a fiber geometry.

cisely, we assume that our domain is the unit cube, i.e., $\Omega = [0,1]^3$. The fibers are defined by connecting the following pairs of points $\{(0, h/2 + i_2 h, h/2 + i_3 h)$, $(1, h/2 + i_2 h, h/2 + i_3 h)\}$, $\{(h/2 + i_1 h, 0, h/2 + i_3 h)$, $(h/2 + i_1 h, 1, h/2 + i_3 h)\}$, and $\{(h/2 + i_1 h, h/2 + i_2 h, 0)$, $(h/2 + i_1 h, h/2 + i_2 h, 1)\}$, for $i_1, i_2, i_3 = 0, 1, \ldots, 1/h - 1$. Here we tacitly assume that $1/h \in \mathbb{N}$. Additionally, we require the diameters of all fibers to be smaller than the side lengths of the subdomains, each of which is assumed to be of equal cubic size and shape. It is evident that the example geometry just described is quite particular. In fact, it can be easily seen that the number of intersections is rather large compared to a random geometry with an equal number of fibers. Despite being artificial we will however see below that this geometry is quite representative in terms of the computational costs of Algorithm 1. Table 1 gives an

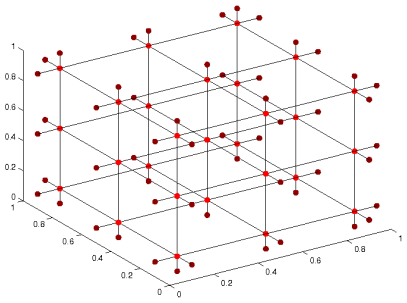


**Fig. 3.** Interior and boundary nodes for a regular fiber structure.

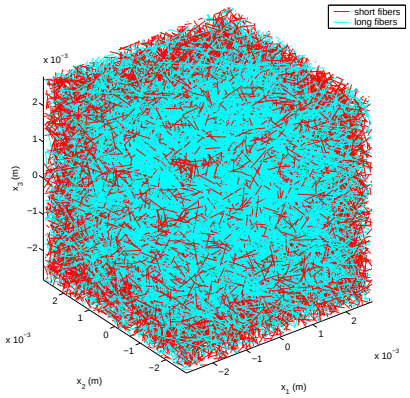



**Fig. 4.** Geometry with 1% svf and equal parts of long and short fibers.

overview of the computational costs of the different steps of Algorithm 1 when applied to this example geometry.

Based on the information in Table 1 we can see that the total numerical complexity of Algorithm 1 (i.e., steps 1-27) is given by

$$\mathcal{O}(n_{\Phi} n_{\Omega,x_1}) + \mathcal{O}\left(\frac{n_{\Phi}^2}{n_{\Omega,x_1}}\right). \tag{3}$$

Thus, we easily deduce that choosing

$$n_{\Omega,x_1} = \mathcal{O}(\sqrt{n_{\Phi}}) \tag{4}$$

leads to a total numerical complexity of

$$\mathcal{O}(n_{\Phi}^{3/2}) \tag{5}$$

when applied to our regular example geometry sketched in Fig. 5. This is of course a major improvement compared to the complexity $\mathcal{O}(n_{\Phi}^2)$ of the standard approach.

*Remark 3.* It should be noted here that the reasoning above is somewhat specific for our example geometry. For general randomly generated fibrous geometries with multiple fiber lengths and diameters we cannot obtain such a nice and compact formula as in (5). Nevertheless, our considerations above are surprisingly representative for more general cases as a collection of examples in section 5 shows.

## 5 Numerical Results and Conclusions

Now, let us take a look at the actual numerical performance of Algorithm 1 when applied to large randomly generated fibrous geometries. In order to do this, we first

| Steps | Order of Complexity | |
|---|---|---|
| 10-13 | $\mathcal{O}(1)$ | |
| 9-15 | $\mathcal{O}(1)$ | Since $\#\hat{\mathscr{J}} \leq 6$. |
| 6-17 | $\mathcal{O}(n_{\Omega,x_1})$ | Since the number of subdomains that each fiber passes through is $\mathcal{O}(n_{\Omega,x_1})$ and each subdomain is checked at most once. Note that $n_{\Omega,x_1} = n_{\Omega,x_2} = n_{\Omega,x_3}$ and that we require the fiber diameters to be smaller than the side lengths of the subdomains. |
| 2-18 | $\mathcal{O}(n_\Phi n_{\Omega,x_1})$ | |
| 20-26 | $\mathcal{O}\left((\#\Phi_{\mathbf{j}})^2\right) = \mathcal{O}\left(\frac{n_\Phi^2}{n_{\Omega,x_1}^4}\right)$ | Since in each subdomain of our regular fiber structure (see Fig. 5) there are $\frac{3}{(hn_{\Omega,x_1})^2}$ fibers and in the entire domain $\Omega$ there are $\frac{3}{h^2}$ fibers, i.e., $n_\Phi = \frac{3}{h^2}$. |
| 19-27 | $\mathcal{O}\left(\#\mathscr{J}\,\frac{n_\Phi^2}{n_{\Omega,x_1}^4}\right) = \mathcal{O}\left(\frac{n_\Phi^2}{n_{\Omega,x_1}}\right)$ | Since $\#\mathscr{J} = n_{\Omega,x_1}^3$ for our cubic domain. |

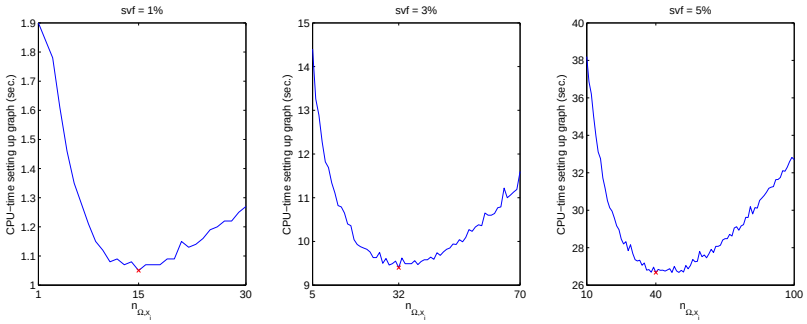
**Table 1.** Computational cost of Algorithm 1

specify the parameters used in the generation of our structures. All geometries are generated by the GeoDict2008 software using a grid of $2000^3$ voxels on $\Omega$, which is chosen to be a cube with side-length 5.6e-3m. Thus, the side-length of a voxel is 2.8e-6m. We consider structures having a solid volume fraction (svf) of 1%, 3%, and 5%, i.e., 1%, 3%, and 5% of $\Omega$ are occupied by fibers, respectively. For each of these svf we consider a geometry with equal parts of infinitely long and short fibers ("short" meaning 100 voxels long), one with infinitely long fibers only, and one with short fibers only. We then consider a series of choices for $n_{\Omega,x_1} = n_{\Omega,x_2} = n_{\Omega,x_3}$ and compare the cpu-times needed for setting up the graphs. To get an impression how these fibrous geometries look we refer to Fig. 4, which shows a plot of the structure with 1% svf and with equal parts of short and long fibers.

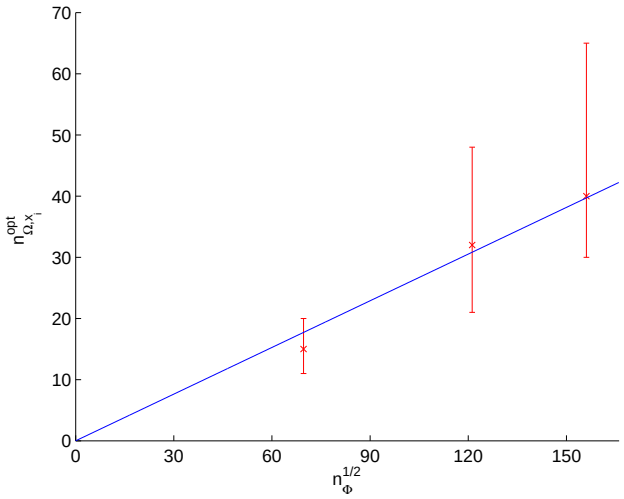
The tables in Figs. 5-7 show the data specific of the problems under consideration (number of fibers, number of nodes, etc.) and the computational costs for the cases $n_{\Omega,x_1} = n_{\Omega,x_2} = n_{\Omega,x_3} = 1$ and $n_{\Omega,x_1} = n_{\Omega,x_1}^{opt}$, where $n_{\Omega,x_1}^{opt}$ is the optimal choice for $n_{\Omega,x_1}$ in terms of the time needed for setting up the graph corresponding to the fibrous geometry. (In order to determine $n_{\Omega,x_1}^{opt}$ we consider a series of $n_{\Omega,x_1}$, see the top plots in Figs. 5-7.)

As we can see, the reduction of cpu-time when choosing $n_{\Omega,x_1} = n_{\Omega,x_1}^{opt}$ instead of $n_{\Omega,x_1} = 1$ is substantial. For the geometries involving only long fibers the time needed for setting up the graph is roughly cut in half (cf. table in Fig. 5). For the fibrous structure with a solid volume fraction (svf) of 5% and only short fibers the cpu-time for constructing the graph is reduced to less than 0.3% when choosing $n_{\Omega,x_1} = n_{\Omega,x_1}^{opt}$ (table in Fig. 6). Looking at the table in 7 we see that also for geometries consisting of short and long fibers the cpu-time for setting up the graph is reduced by more than one order of magnitude when choosing the optimal $n_{\Omega,x_1}$.

For the instances that we consider we see that by choosing $n_{\Omega,x_1} = n_{\Omega,x_1}^{opt}$ the computational cost of constructing the graph corresponding to our geometry can be re-




CPU-times needed for setting up the graph for different choices of $n_{\Omega,x_1} = n_{\Omega,x_2} = n_{\Omega,x_3}$.




Optimal choice of $n_{\Omega,x_1}$, i.e. $n^{opt}_{\Omega,x_1}$ (x), vs. $\sqrt{n_\Phi}$ with 5% deviation margins and a linear leasts-squares fit.

| svf short fibers | 0% | | 0% | | 0% | |
|---|---|---|---|---|---|---|
| svf long fiber | 1% | | 3% | | 5% | |
| # fibers | 4851 | | 14686 | | 24366 | |
| # interior nodes | 15462 | | 132121 | | 351366 | |
| effective conductivity tensor | 2.64e-2 - - - 2.65e-2 - - - 2.64e-2 | | 3.13e-2 - - - 3.14e-2 - - - 3.15e-2 | | 3.66e-2 - - - 3.64e-2 - - - 3.64e-2 | |
| # coarse grid cells | 1 | $15^3$ | 1 | $32^3$ | 1 | $40^3$ |
| total CPU-time (sec.) | 2.3e0 | 1.4e0 | 2.9e1 | 1.7e1 | 1.1e2 | 6.3e1 |
| CPU-time constructing the graph | 1.9e0 | 1.1e0 | 1.8e1 | 9.4e0 | 5.7e1 | 2.7e1 |
| CPU-time solving the system | $< 1$ | $< 1$ | 1.1e1 | 7.6e0 | 5.4e1 | 3.5e1 |

Computational results and costs.

**Fig. 5.** CPU-time analysis and numerical results for geometries with **only long** fibers and solid volume fractions of 1%, 3%, and 5%, respectively.

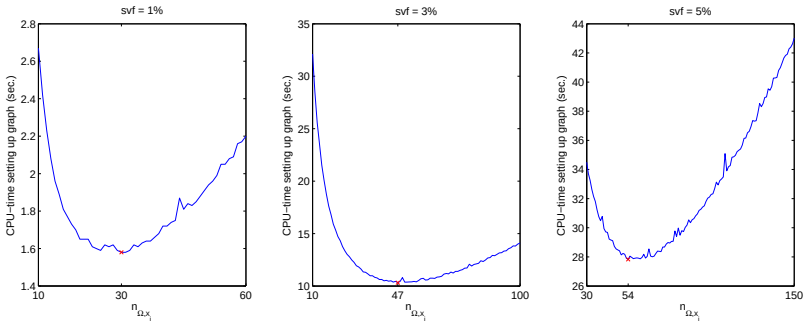
duced to the same order of magnitude as the cost needed for solving the arising linear system. (Here we would like to remark that for solving the linear system we employ the ILU preconditioned Conjugate Gradient (CG) solver implemented in the LAS-Pack package (see http://www.mgnet.org/mgnet/codes/laspack/html/laspack.html) using a relative residual reduction of $1e-6$ as stopping criterion.) Before, i.e., when choosing $n_{\Omega,x_1} = 1$, almost the entire computational cost for determining an approximation to the effective thermal conductivity tensor was devoted to setting up the computational graph. Therefore, it was not feasible to spend much effort on speeding up the solution of the arising linear system. Now, with this new approach of dividing $\Omega$ into subdomains, we see that in some cases the cpu-time for solving the arising linear system can actually exceed the cpu-time for constructing the graph (cf. tables in Figs. 5-7). With this observation it seems reasonable to also optimize the process of solving the arising linear system - e.g. by employing algebraic multi-grid methods and the like - which is a topic of our further research.

As an interesting side note we would like to remark that in all investigated cases (cf. tables in Figs. 5-7) the cpu-time for solving the linear system also reduces (by around 30%) when choosing the optimal $n_{\Omega,x_1}$. This observation seems surprising, since the graph constructed by Algorithm 1 and the number of CG-iterations required to satisfy the convergence criterion are independent of the choice for $n_{\Omega,x_1}$. The only plausible explanation that we have for this certainly desirable side effect is that for $n_{\Omega,x_1} = n^{opt}_{\Omega,x_1}$ the nodes of the graph are not in the same order as when choosing $n_{\Omega,x_1} = 1$. Apparently, this re-odering of the unknowns speeds up the matrix-vector multiplication of the system matrix, which could be due to a better cache-optimization. Providing a detailed analysis of this issue is, however, beyond the scope of this article.

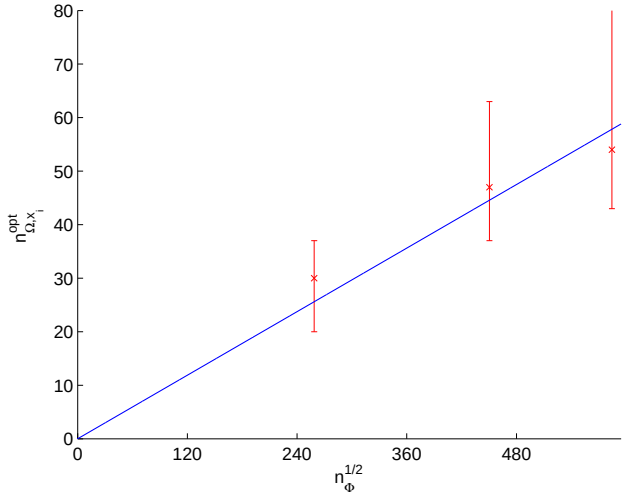
Looking at the graphs in Figs. 5-7, where the cpu-time for constructing the graph is plotted vs. the choice for $n_{\Omega,x_1}$, we see that there is in fact an optimal choice $n^{opt}_{\Omega,x_1}$. This observation can be explained via (3). When choosing $n_{\Omega,x_1}$ larger (smaller) than $n^{opt}_{\Omega,x_1}$ the first (second) term of (3) dominates.

Now, we would like to investigate the question, whether relation (4), which we derived for the very regular fiber structure shown in Fig. 5, also holds - at least approximately - for our randomly generated geometries. For this we plot $n^{opt}_{\Omega,x_1}$ against $\sqrt{n_\Phi}$ for different fibrous geometries (see lower left plots in Figs. 5-7). Of course, we can only hope for (4) to hold for structures with different solid volume fractions but with the same kind of fibers. Therefore, we only try to verify (4) for these cases. Looking at the least squares linear fit (blue line) in Figs. 5-7, where the fitted line is forced through the origin and thus the only free parameter is its slope, we can see that (4) is indeed quite well satisfied. Nevertheless, the constant involved in (4) is different for different choices of fibers. For the sequence of geometries with svf 1%, 3%, and 5% and only long fibers it is approximated to 2.54e-1, while for the cases of only short fibers it is approximately 9.89e-2. The constant for the geometries involving equal parts of long and short fibers is estimated to 1.11e-1 and thus in-between the two former ones.

In addition to $n^{opt}_{\Omega,x_1}$ the lower left plots in Figs. 5-7 also show margins which correspond to those choices for $n_{\Omega,x_1}$ for which the cpu-time for setting up the graph




CPU-times needed for setting up the graph for different choices of $n_{\Omega,x_1} = n_{\Omega,x_2} = n_{\Omega,x_3}$.

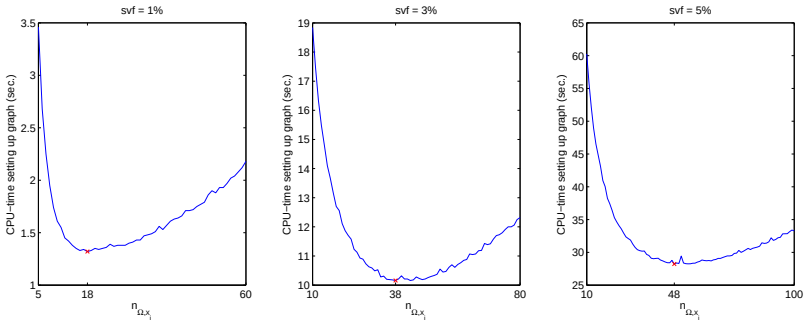



Optimal choice of $n_{\Omega,x_1}$, i.e. $n^{opt}_{\Omega,x_1}$ (x), vs. $\sqrt{n_\Phi}$ with 5% deviation margins and a linear least-squares fit.

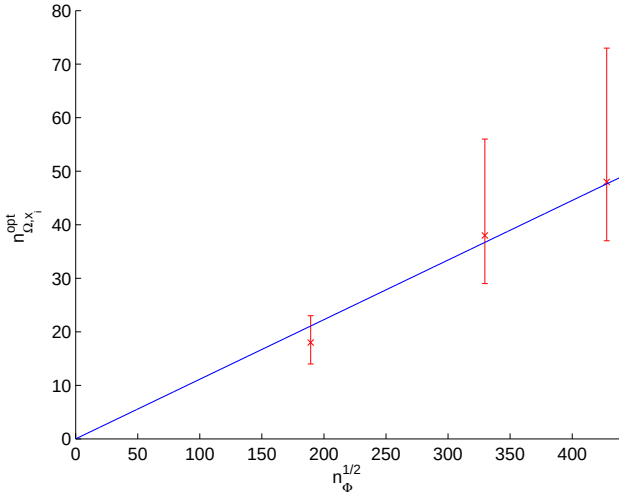
| svf short fibers | 1% | | 3% | | 5% | |
|---|---|---|---|---|---|---|
| svf long fiber | 0% | | 0% | | 0% | |
| # fibers | 66953 | | 202845 | | 341543 | |
| # interior nodes | 2127 | | 113936 | | 378634 | |
| effective conductivity tensor | 2.40e-2 - - - 2.40e-2 - - - 2.40e-2 | | 2.40e-2 - - - 2.40e-2 - - - 2.40e-2 | | 2.57e-2 - - - 2.57e-2 - - - 2.57e-2 | |
| # coarse grid cells | 1 | $15^3$ | 1 | $32^3$ | 1 | $40^3$ |
| total CPU-time (sec.) | 3.6e2 | 2.2e0 | 3.5e3 | 5.9e1 | 9.9e3 | 1.1e2 |
| CPU-time constructing the graph | 3.6e2 | 1.6e0 | 3.4e3 | 1.0e1 | 9.8e3 | 2.8e1 |
| CPU-time solving the system | $< 1$ | $< 1$ | 6.5e1 | 4.7e1 | 1.1e2 | 7.7e1 |

Computational results and costs.

**Fig. 6.** CPU-time analysis and numerical results for geometries with **only short** fibers and solid volume fractions of 1%, 3%, and 5%, respectively.




CPU-times needed for setting up the graph for different choices of $n_{\Omega,x_1} = n_{\Omega,x_2} = n_{\Omega,x_3}$.




Optimal choice of $n_{\Omega,x_1}$, i.e. $n^{opt}_{\Omega,x_1}$ (x), vs. $\sqrt{n_\Phi}$ with 5% deviation margins and a linear leasts-squares fit.

| svf short fibers | 0.5% | | 1.5% | | 2.5% | |
|---|---|---|---|---|---|---|
| svf long fiber | 0.5% | | 1.5% | | 2.5% | |
| # fibers | 35830 | | 108668 | | 182974 | |
| # interior nodes | 14383 | | 136086 | | 366286 | |
| effective conductivity tensor | 2.52e-2 - - - 2.52e-2 - - - 2.52e-2 | | 2.80e-2 - - - 2.79e-2 - - - 2.79e-2 | | 3.14e-2 - - - 3.14e-2 - - - 3.15e-2 | |
| # coarse grid cells | 1 | $18^3$ | 1 | $38^3$ | 1 | $48^3$ |
| total CPU-time (sec.) | 9.5e1 | 2.0e0 | 1.0e3 | 2.1e1 | 2.9e3 | 7.1e1 |
| CPU-time constructing the graph | 9.4e1 | 1.3e0 | 9.9e2 | 1.0e1 | 2.8e3 | 2.8e1 |
| CPU-time solving the system | $< 1$ | $< 1$ | 1.4e1 | 1.0e1 | 6.4e1 | 4.1e1 |

Computational results and costs.

**Fig. 7.** CPU-time analysis and numerical results for geometries with with **equal parts of short and long** fibers and solid volume fractions of 1%, 3%, and 5%, respectively.

is at most 5% higher than for $n^{opt}_{\Omega,x_1}$. For practical problems it of course doesn't make sense to apply Algorithm 1 for several choices of $n_{\Omega,x_1}$ to determine the optimal one. Instead one is interested in approximating $n^{opt}_{\Omega,x_1}$ beforehand, and then use this approximation in the calculations. It is quite obvious that (4) can be used to predict an approximation to $n^{opt}_{\Omega,x_1}$. Furthermore, it should be noted that the margins shown in 5-7 indicate that - especially for large and thus costly geometries - one doesn't really have to approximate $n^{opt}_{\Omega,x_1}$ very accurately in order to obtain almost optimal performance. Thus, it seems promising that an automatic way of approximating $n^{opt}_{\Omega,x_1}$, which could then be used in Algorithm 1, can be implemented. This is also an objective of our further research.

On the whole, we would like to conclude that Algorithm 1 constitutes a very powerful enhancement of the approach presented in [2]. The computational costs are significantly reduced, which makes our graph-laplacian approach applicable to even larger geometries containing even more fibers.

## References

[1] Ewing, R., Iliev, O., Lazarov, R., Rybak, I., Willems, J.: An efficient approach for upscaling properties of composite materials with high contrast of coefficients. Technical Report 132, Fraunhofer ITWM, 2007.

[2] Iliev, O., Lazarov, R., Willems, J.: A graph-laplacian approach for calculating the effective thermal conductivity of complicated fiber geometries. Technical Report 142, Fraunhofer ITWM, 2008.

[3] Jikov, V.V., Kozlov, S.M., Oleinik, O.A.: *Homogenization of Differential Operators and Integral Functionals*. Springer, 1st ed., 1994.