Analysis of Mesh Strategies for Rapid Source Location in Chemical/Biological Attacks

by

Patricia A. Howard

A Thesis

Submitted to the Faculty

of

WORCESTER POLYTECHNIC INSTITUTE

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

in

Applied Mathematics

by

May 2004

APPROVED:

Dr. Suzanne L. Weekes, Thesis Advisor

Dr. Bogdan Vernescu, Department Head

Abstract

Currently, researchers at Sandia National Laboratories are creating software that is designed to determine the source of a toxic release given sensor readings of the toxin concentration at fixed locations in the building. One of the most important concerns in solving such problems is computation time since even a crude approximation to the source, if found in a timely manner, will give emergency personnel the chance to take appropriate actions to contain the substance.

The manner in which the toxin spreads depends on the air flow within the building. Due to the turbulence in the air flow, it is necessary to calculate the flow field on a fine mesh. Unfortunately, using a fine mesh for every calculation in this problem may result in prohibitively long computation times when other features are incorporated into the model. The goal of this thesis is to reduce the computation time required by the software mentioned above by applying two different mesh coarsening strategies after the flow field is computed. The first of these strategies is to use a uniformly coarse mesh and the second is to use our knowledge of the air flow in the building to construct an adaptive mesh. The objective of the latter strategy is to use a fine mesh only in areas where it is absolutely necessary, i.e., in areas where there is a great change in the flow field.

Acknowledgments

A friend once told me our expectations for how long a particular task will take often fall short of what happens in reality. He always went with the approach of multiplying that estimate by three and adding a day. I guess as humans, or maybe idealists, we like to forget that simple solutions are often the hardest to find and the details of seemingly simple methods are complex. Fortunately, a number of people have been there to help make my reminder of these subtleties a little less painful.

Since it is the questions that we cannot answer that make us strive to improve ourselves and our understanding, I would like to thank Professor Homer Walker and my advisor Professor Suzanne Weekes. Without their questions, I probably would not have come as far as I have in my understanding.

I would also like to recognize a number of researchers at Sandia National Laboratories without whose assistance I never would have had the opportunity to work on a project of this magnitude. Since many aspects of fluid mechanics are still a mystery to me, I am grateful there are people like Steve Margolis who are capable of mastering the finer points. In addition, without the help of Kevin Long, and to a lesser degree Jonathan Hu and James Willenbring, I likely never would have survived the trials and tribulations of installing software. Also, I would like to thank Philippe Pebay for his input concerning BAMG. Most importantly though, I would like to thank Paul Boggs. In addition to helping me understand the finer points of constrained optimization, I have found Paul's guidance and encouragement to be invaluable.

Lastly I would like to thank my friends and family, while they rarely understood what I was trying to do, at least they tried and for that I am grateful.

Contents

Chapter 1

Introduction

Suppose a lethal and highly contagious virus is released in an international airport with no means of detection. Days would pass before infected patients sought medical treatment. By that time, hundreds of thousands of people throughout the world would be infected and tracing the virus back to the source would be nearly impossible. Furthermore, even if the original source were determined, any evidence that would indicate who initiated the act would have likely been destroyed. Now, suppose that the same airport had been equipped with some means of detecting and locating the source of such a release. Given knowledge of this source location in a timely fashion, emergency personnel would be able to take appropriate measures to prevent further spread of the virus, thereby significantly reducing fatalities. This scenario applies not only to viruses released in airports but also to any biological or chemical toxin released in a heavily populated building.

To locate the source of a toxic release similar to the one described above, it is necessary to have the following three pieces of technology in place beforehand. First, we need to have sensors that accurately determine the type and concentration of the toxin; second, it is necessary to have knowledge of the flow field, and lastly we need

software that can rapidly determine the source of the toxin given the concentration readings from these sensors. In this thesis, we will only discuss the third problem. Note that while a certain amount of accuracy is important, it is absolutely critical that at least a crude approximation to the source location be determined quickly. Therefore, a good software implementation should, given moderate computing ability, locate the source within a few meters in a short period of time.

Our approach for determining the concentration and source location of the toxin is to minimize the discrepancy between the sensor readings and the computed concentration subject to a model of how the toxin permeates the building. We will refer to this optimization problem as the source inversion problem. It is not difficult to see that the manner in which the toxin spreads depends on the air flow within the building. This air flow is determined by the building's heating, ventilating and air-conditioning (HVAC) system. It is our understanding that in a real world scenario similar to the one which we described, the air flow will not change considerably throughout the day. Therefore, we assume that only a handful of flow fields are necessary to effectively describe the air flow in the building. Thus, each of these flow fields may be calculated beforehand and stored for use when a toxin is detected. The approach briefly outlined above has already been implemented at Sandia National Laboratories and will be written up in [5].

Due to the complexity of the flow in the types of buildings we are considering, it is necessary to calculate the flow field on a relatively fine mesh. Since we are not tremendously concerned with how quickly the flow field is calculated, computation time is not of particular concern for that part of the problem. Unfortunately, using the same mesh for the source inversion problem restricts how quickly we can determine the solution.

The goal of this thesis is to investigate the effect of coarsening the mesh on the computation time required to solve the source inversion problem and the accuracy of

our solutions. In light of this goal, we consider two different strategies for coarsening the mesh on which the flow field is provided. The first of these strategies is to project the flow field onto a uniformly coarse mesh that we can then use to solve the source inversion problem. The second is to use an adaptive mesh approach in which the coarseness of the mesh in a particular area of the room is determined by how the flow is changing in that area.

Finally, while it is clear that this problem is time-dependent, we believe that much can be learned from the simpler steady-state case. Therefore, we only consider the effect of these mesh refining strategies on the steady-state case. In addition, we only consider the two-dimensional problem here.

The remainder of this thesis is organized in the following manner. In Chapter 2, we give a brief description of some of the important features of the flow model. In Chapter 3, we describe the numerical methods we use to solve the steady-state source inversion problem which is developed in Chapter 4. The strategies and software that we use to create the various meshes for this work are then discussed in Chapter 5. Finally, the numerical results for this work are presented in Chapter 6 and a discussion of future work is given in Chapter 7.

Chapter 2

Flow Field Model

The spread of the toxin depends greatly on the air flow in the building. Therefore, we provide a brief description of some of the important features of the flow field model here.

If we assume that the concentration of the toxin is insufficient to affect the flow field then we may use the time-dependent continuity and momentum (Navier-Stokes) equations for incompressible fluid flow given by

$$\begin{aligned} \overrightarrow{\nabla} \cdot \mathbf{u} &= 0 \\ \tfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \overrightarrow{\nabla})\mathbf{u} + \tfrac{1}{\rho}\overrightarrow{\nabla} p - \nu \nabla^2 \mathbf{u} &= 0, \end{aligned} \tag{2.1}$$

where the arrow over the gradient operator indicates the location of the object to which the operation is applied. Here, the dependent variables $\mathbf{u}$ and p are, respectively, fluid velocity and pressure, which are both functions of space and time. In addition, ρ and ν are respectively, fluid density and kinematic viscosity, which are both assumed to be constant. It is important to note that the types of buildings we are considering do not have simple air flow behavior. In particular, the value of the Reynolds number for these types of problems is large, resulting in inherent instability in the air flow particularly in the vicinity of boundary layers. In addition, the

presence of people, opening and closing doors, counters, chairs, and even the size of the rooms involved contribute to the complexity of the air flow. Consequently, it is necessary to incorporate turbulence into the above model.

The approach applied in [5] to account for this behavior involves decomposing the dependent variables into time-averaged and fluctuating components. This results in the well-known Reynolds-averaged Navier-Stokes (RANS) equations, given by

$$\begin{aligned} \overrightarrow{\nabla} \cdot \mathbf{U} &= 0 \\ (\mathbf{U} \cdot \overrightarrow{\nabla})\mathbf{U} + \tfrac{1}{\rho}\overrightarrow{\nabla} P - \nu \nabla^2 \mathbf{U} - \tfrac{1}{\rho}(\underline{\underline{\tau}} \cdot \overleftarrow{\nabla}) &= 0, \end{aligned} \tag{2.2}$$

where $\mathbf{U}$ and P are the time-averaged components of $\mathbf{u}$ and p respectively. Also, $\underline{\underline{\tau}}$ is the Reynolds stress tensor given by

$$\underline{\underline{\tau}} = [\tau_{ij}] = [-\rho \overline{u_i' u_j'}], \tag{2.3}$$

where u_i' denotes the fluctuating component of u_i and

$$\overline{v} = \overline{v}(t_0) = \lim_{\gamma \to \infty} \frac{1}{\gamma} \int_{t_0}^{\gamma + t_0} v \; dt. \tag{2.4}$$

Another consequence of the turbulent behavior in the air flow is that it is necessary to solve the system (2.2) on a fine mesh. Fortunately, the overall air flow in the types of buildings we are considering does not change considerably over the course of a day. Thus, only a handful of different flow fields are required to adequately represent the air flow at any given time during any given season. As a result, the flow field can be computed beforehand and stored for later use. Therefore, the long computation times that result from the need for a fine mesh are of less concern at this stage. Unfortunately, solving the source inversion problem on this same fine mesh results in computation times that are unnecessarily large. Thus, when we discuss the issue of coarsening the mesh, we must also address the problem of projecting the flow field onto this mesh; see Chapter 5.

Chapter 3

Numerical Methods for Constrained Optimization

The source inversion problems to be presented in subsequent chapters are formulated as optimization problems of the following form

$$\begin{aligned} \underset{\mathbf{y}}{\text{minimize}} \quad & \mathbf{y}^{\mathrm{T}}\mathrm{Q}\mathbf{y} + \mathbf{g}^{\mathrm{T}}\mathbf{y} \\ \text{subject to} \quad & \mathbf{a}_i\mathbf{y} + b_i = 0, \text{ for } i \in \mathcal{E}, \\ & \mathbf{a}_i\mathbf{y} + b_i \leq 0, \text{ for } i \in \mathcal{I}, \end{aligned} \tag{3.1}$$

where $\mathbf{y}$, $\mathbf{g}$ and $\mathbf{a}_i$ are a vectors with n components, Q is an $n \times n$ positive definite matrix, b_i is a scalar and $\mathcal{E}$ and $\mathcal{I}$ are sets of indices corresponding to the equality and inequality constraints, respectively. Problems of this type are referred to as convex quadratic programs (QP) where the convexity is due to the positive definiteness of Q. In this chapter, we begin by discussing some theory that is necessary for solving (3.1). Then, we discuss methods for solving the equality constrained version of this problem, i.e. where $\mathcal{I} = \emptyset$, before addressing the more general case. Methods for both of these situations will be relevant in later chapters.

3.1 Theory for Convex Quadratic Programs

Consider the optimization problem

$$\begin{aligned} \underset{\mathbf{y}}{\text{minimize}} \quad & f(\mathbf{y}) \\ \text{subject to} \quad & l_i(\mathbf{y}) = 0, \text{ for } i \in \mathcal{E}, \\ & l_i(\mathbf{y}) \leq 0, \text{ for } i \in \mathcal{I}, \end{aligned} \tag{3.2}$$

where $f(\mathbf{y})$ is referred to as the objective function of the optimization problem. Let $\mathbf{y}$ be a feasible point, i.e., a point satisfying all the constraints of (3.2). Then $\mathbf{y}$ is a local solution of (3.1) if there is a neighborhood N such that for all feasible points $\overline{\mathbf{y}} \in N$ we have $f(\mathbf{y}) \leq f(\overline{\mathbf{y}})$. Consequently, if a point $\mathbf{y}$ is a local solution, then no vector $\mathbf{d} \in N$ exists such that $\overline{\mathbf{y}} = \mathbf{y} + \mathbf{d}$ is a feasible point and $\nabla f(\mathbf{y})^{\mathrm{T}}\mathbf{d} < 0$. Notice that $\nabla f(\mathbf{y})^{\mathrm{T}}\mathbf{d} < 0$ implies that $\mathbf{d}$ is a descent direction of the objective function, $f(\mathbf{y})$. Suppose that $\mathbf{y}$ is a local solution. For a general constraint, $l_i(\mathbf{y}) = 0$ or $l_i(\mathbf{y}) \leq 0$, we know that $\nabla l_i(\mathbf{y})$ is perpendicular to the level sets of $l_i(\mathbf{y})$. Therefore, if $|\mathcal{E}| = 1$ then $\nabla f(\mathbf{y})$ and $\nabla l_1(\mathbf{y})$ must be parallel. As shown in Figure 3.1, if $\nabla f(\mathbf{y})$ and $\nabla l_1(\mathbf{y})$ are not parallel then there exists a descent direction $\mathbf{d}$ such that $\mathbf{d} + \mathbf{y}$ is feasible. On the

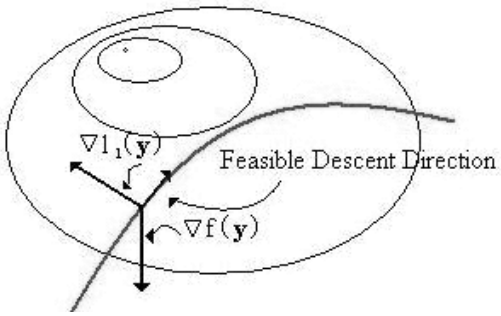

Figure 3.1: Feasible descent direction for an equality constrained problem with $\mathcal{E} = 1$.

other hand if $|\mathcal{E}| > 1$ then $\nabla f(\mathbf{y})$ must be a linear combination of the vectors $\nabla l_i(\mathbf{y})$ for $i \in \mathcal{E}$; see [10]. Due to similar reasoning, if $|\mathcal{I}| = 1$ then $\nabla f(\mathbf{y})$ and $\nabla l_1(\mathbf{y})$ must be parallel and pointing in opposite directions. For the case where $|\mathcal{I}| > 1$, $\nabla f(\mathbf{y})$

must be a linear combination of the vectors $\nabla l_i(\mathbf{y})$, $i \in \mathcal{I}$, with negative coefficients; see Figure 3.2. To state this more clearly, if $\mathbf{y}$ is a local solution then

$$\nabla f(\mathbf{y}) = \sum_{i \in \mathcal{E}} \lambda_i \nabla l_i(\mathbf{y}), \tag{3.3}$$

where $\lambda_i \leq 0$ for all $i \in \mathcal{I}$.

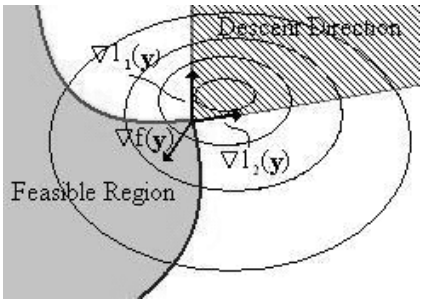

Figure 3.2: At a local solution of an inequality constrained optimization problem $\nabla f(\mathbf{y})$ must be a linear combination of the vectors $\nabla l_i(\mathbf{y})$ for $i \in \mathcal{I}$ with negative coefficients.

The observations above can be stated formally for convex QP problems as the following theorem.

Theorem 3.1 *A point $\mathbf{y}^*$ is a local solution to a convex QP defined by (3.1) if and only if there exists a vector λ^*, with components λ_i^*, $i \in \mathcal{E} \cup \mathcal{I}$, such that the following conditions, referred to as the Karush-Kuhn-Tucker (KKT) conditions, are satisfied:*

$$\begin{aligned}
&\nabla_{\mathbf{y}} \mathcal{L}(\mathbf{y}^*,\ \lambda^*) = 0, \\
&\mathbf{a}_i \mathbf{y}^* + b_i = 0,\ \text{for } i \in \mathcal{E}, \\
&\mathbf{a}_i \mathbf{y}^* + b_i \leq 0,\ \text{for } i \in \mathcal{I}, \\
&\lambda_i^* \leq 0,\ \text{for } i \in \mathcal{I}, \\
&\lambda_i^* (\mathbf{a}_i \mathbf{y}^* + b_i) = 0,\ \text{for } i \in \mathcal{I}.
\end{aligned} \tag{3.4}$$

In addition, if the Hessian of the objective function, i.e., Q, is positive definite then $\mathbf{y}^*$ is a unique solution.

Here, the Lagrangian $\mathcal{L}(\mathbf{y},\ \lambda)$ is defined by,

$$\mathcal{L}(\mathbf{y},\ \lambda) = \mathbf{y}^{\mathrm{T}}\mathrm{Q}\mathbf{y} + \mathbf{g}^{\mathrm{T}}\mathbf{y} - \sum_{i \in \mathcal{A}(\mathbf{y})} \lambda_i(\mathbf{a}_i\mathbf{y} + b_i), \tag{3.5}$$

where $\mathcal{A}(\mathbf{y})$ is the active set at $\mathbf{y}$ defined by

$$\mathcal{A}(\mathbf{y}) = \{i \in \mathcal{E} \cup \mathcal{I} \mid \mathbf{a}_i\mathbf{y} + b_i = 0\}. \tag{3.6}$$

See [10] for a proof of Theorem 3.1.

3.2 Equality Constrained Convex QPs

For the equality constrained case, i.e., when $\mathcal{I} = \emptyset$, the active set at any feasible point is equivalent to the set of indices corresponding to the equality constraints. Therefore, by applying Theorem 3.1, finding a local solution to (3.1) translates into the problem of solving the following system of equations for $\mathbf{y}$ and λ_i, $i \in \mathcal{E}$,

$$\nabla_{\mathbf{y}}\mathcal{L}(\mathbf{y}, \lambda) = 0, \tag{3.7}$$

$$\mathbf{a}_i\mathbf{y} + b_i = 0, \text{ for all } i \in \mathcal{E}. \tag{3.8}$$

Notice that the second of these equations (3.8) is equivalent to $\nabla_{\lambda}\mathcal{L}(\mathbf{y}, \lambda) = \mathbf{0}$.

The system given by (3.7) and (3.8) can be solved using any number of linear solvers. The details are beyond the scope of this thesis.

3.3 Inequality Constrained Convex QPs

The quadratic programming problem (3.1) becomes more difficult to solve when inequality constraints are present, i.e., $\mathcal{I} \neq \emptyset$. If we know the active set at the desired solution, then the problem can be reformulated as an equality constrained convex QP

and solved as discussed in the previous section. Unfortunately, complete knowledge of the active set at the solution is rare. As a result, more complicated algorithms are required to solve a general convex QP.

For the algorithm we use, we assume the optimization problem contains only inequality constraints. Thus, each equality constraint can be either represented as a pair of inequality constraints or incorporated into the objective function in some manner. For inequality constrained problems in this work, we incorporate the equality constraints into the objective function via a quadratic penalty term resulting in the following modified optimization problem

$$\begin{aligned} \underset{\mathbf{y}}{\text{minimize}} \quad & \mathbf{y}^{\mathrm{T}}\mathrm{Q}\mathbf{y} + \mathbf{g}^{\mathrm{T}}\mathbf{y} + \tfrac{1}{2}\sigma||\mathrm{A}_{\mathcal{E}}\mathbf{y} + \mathbf{b}_{\mathcal{E}}||_2^2 \\ \text{subject to } & \mathrm{A}_{\mathcal{I}}\mathbf{y} + \mathbf{b}_{\mathcal{I}} \leq 0, \end{aligned} \tag{3.9}$$

where $\sigma > 0$ is called the penalty parameter, $\mathrm{A}_{\mathcal{E}} = [\mathbf{a}_i]_{i\in\mathcal{E}}^{\mathrm{T}}$ and $\mathbf{b}_{\mathcal{E}} = [b_i]_{i\in\mathcal{E}}$ with $\mathrm{A}_{\mathcal{I}}$ and $\mathbf{b}_{\mathcal{I}}$ defined similarly. In order to solve the penalty form, (3.9), we gradually increase σ until $||\mathrm{A}_{\mathcal{E}}\mathbf{y}+\mathbf{b}_{\mathcal{E}}||_\infty$ is sufficiently small. This idea is incorporated into each iteration of the inequality constrained QP solver we describe next.

We rewrite (3.9) in the form

$$\begin{aligned} \underset{\mathbf{y}}{\text{minimize}} \quad & \tfrac{1}{2}\mathbf{y}^{\mathrm{T}}\overline{\mathrm{Q}}\mathbf{y} + \overline{\mathbf{g}}^{\mathrm{T}}\mathbf{y} \\ \text{subject to } & \mathrm{A}_{\mathcal{I}}\mathbf{y} + \mathbf{b}_{\mathcal{I}} \leq 0, \end{aligned} \tag{3.10}$$

where $\overline{\mathbf{g}}$ and $\mathbf{y}$ are vectors with n components, $\overline{\mathrm{Q}}$ is an $n \times n$ symmetric positive definite matrix, $\mathbf{b}_{\mathcal{I}}$ is a vector with m components, and $\mathrm{A}_{\mathcal{I}}$ in an $m \times n$ matrix with $m = |\mathcal{I}|$. Note that $\overline{\mathrm{Q}}$ depends on σ.

To solve (3.10), we apply a primal, interior point method called O3D, which stands for Optimizing over 3-Dimensional subspaces. This algorithm is implemented in a software package produced by Sandia National Laboratories. We provide a brief description of some of the relevant aspects of this algorithm here and direct the

reader to [2] for further details and [3] and [4] for an extension of this algorithm to solving sequential quadratic programming problems.

As one might surmise from the name of the algorithm, at each iteration, O3D solves a subproblem consisting of the original problem, (3.10), restricted to a three-dimensional subspace. Solving this subproblem results in a search direction. The approximate solution to (3.10) at the current iteration is then found by following this search direction to either the minimizer of the subproblem, i.e., the full length of the direction is used, or a point that is 99% of the distance to the boundary of the feasible region along that search direction. Note that this method requires a strictly feasible starting point, i.e., a point such that $\mathrm{A}_{\mathcal{I}}\mathbf{y} + \mathbf{b}_{\mathcal{I}} < 0$. For this work, a strictly feasible starting point is obtained by slightly altering the solution obtained by solving a similar equality constrained problem; see Chapter 4. The algorithm may be described more explicitly as follows.

Algorithm 3.1 *(O3D)*

Given a strictly feasible starting point $\mathbf{y}_0$, $\epsilon > 0$, $z > 0$ and $\sigma_0 > 0$

For $j = 0, 1, 2, ...$

Let ζ^* be the solution to

$$\begin{aligned} \underset{\zeta}{\text{minimize}} \quad & \tfrac{1}{2}(\mathbf{y}_j + \mathcal{P}\zeta)^{\mathrm{T}}\overline{\mathrm{Q}}(\mathbf{y}_j + \mathcal{P}\zeta) + \overline{\mathbf{g}}^{\mathrm{T}}(\mathbf{y}_j + \mathcal{P}\zeta) \\ \text{subject to} \quad & \mathrm{A}_{\mathcal{I}}(\mathbf{y}_j + \mathcal{P}\zeta) + \mathbf{b}_{\mathcal{I}} \leq 0, \text{ for } i \in \mathcal{I}, \end{aligned} \tag{3.11}$$

where $\mathcal{P}$ is an $n \times 3$ matrix of search directions and $\zeta \in \Re^3$

$\mathbf{y}_{j+1} \leftarrow \mathbf{y}_{j+1} + \mu\mathcal{P}\zeta$

If final convergence criteria are satisfied

STOP

Else if $||\mathrm{A}_{\mathcal{E}}\mathbf{y} + \mathbf{b}_{\mathcal{E}}||_\infty < \epsilon$ for all $i \in \mathcal{E}$

$\sigma_{j+1} = \sigma_j + z$

Else

$\sigma_{j+1} = \sigma_j$

End

The search directions, $\mathbf{p}_i$, that form the columns of $\mathcal{P}$ are solutions to

$$\left[\mathrm{A}_{\mathcal{I}}^{\mathrm{T}}\mathrm{D}^2\mathrm{A}_{\mathcal{I}} + \frac{\overline{\mathrm{Q}}}{\eta}\right]\mathbf{p}_i = \mathbf{t}_i \text{ for } i = 1, 2, 3, \tag{3.12}$$

where η is a scalar that depends upon the current iterate, $\mathbf{y}_j$, D is a diagonal matrix with entries given by

$$d_{ll} = -\frac{1}{(\mathrm{A}_{\mathcal{I}}\mathbf{y}_j + \mathbf{b}_{\mathcal{I}})_l} \text{ for } l = 1, 2, ..., m \tag{3.13}$$

and $\mathbf{t}_i$ is chosen so that one of these search directions is always a descent direction with respect to the objective function. Also, if we let $\overline{\mu}$ be the step length such that $\mathbf{y}_{j+1}$ is 99% of the distance from $\mathbf{y}_j$ to the boundary of the feasible region along the search direction $\mathcal{P}\zeta$, then the step length μ in Algorithm 3.1 is defined by

$$\mu = \min(1, \overline{\mu}). \tag{3.14}$$

The subproblem (3.11) is solved via the dual affine method given by the following algorithm.

Algorithm 3.2 *(Dual Affine Method)*

Given a strictly feasible starting point ζ_0

For $k = 0, 1, 2, ...$

Solve the 3×3 system

$$((\mathrm{A}_{\mathcal{I}}\mathcal{P})^{\mathrm{T}}\overline{\mathrm{D}}^2(\mathrm{A}_{\mathcal{I}}\mathcal{P}) + \mathcal{P}^{\mathrm{T}}\overline{\mathrm{Q}}\mathcal{P})\mathbf{w} = (\overline{\mathbf{g}}^{\mathrm{T}}\mathcal{P} + 2\mathbf{y}_j^{\mathrm{T}}\mathcal{P}) + \mathcal{P}^{\mathrm{T}}\overline{\mathrm{Q}}\mathcal{P}\zeta_k, \tag{3.15}$$

where $\overline{\mathrm{D}}$ is a diagonal matrix with $\overline{d_{ll}} = -\dfrac{1}{\mathrm{A}_{\mathcal{I}}\mathcal{P}\zeta_k + \mathrm{A}_{\mathcal{I}}\mathbf{y}_j + \mathbf{b}_{\mathcal{I}}}$.

$\zeta_{k+1} \leftarrow \zeta_k + \mu\mathbf{w}$, where μ is defined in the same manner as in Algorithm 3.1.

If the final convergence criteria are satisfied

STOP

End

Chapter 4

The Steady-State Source Inversion Problem

The source inversion problem is formulated as an optimization problem that is constrained by the movement of the toxin through the building. This movement depends on the flow field. We begin this chapter by discussing the model of the toxin's movement that will appear in [5], referred to as the toxin transport model. Then we discuss a few possible formulations of the source inversion problem.

4.1 Toxin Transport Model

If we assume that the time-averaged flow field from Chapter 2 is given, then we can write the following continuity equation

$$k\nabla^2 c - (\mathbf{U} \cdot \nabla)c + \nabla \cdot \mathcal{J} + s = 0 \tag{4.1}$$

where k is the diffusivity constant and the dependent variables c and s are time-averaged components which are both dependent on space. Here, c is the concentration of the toxin present in the building and s represents the concentration of the toxin

we are releasing into the building which we refer to as the source. In addition, $\mathcal{J}$ is the turbulence mass flux vector which is defined by

$$\mathcal{J} = [\mathcal{J}_i] = [-\overline{u_i' c'}], \tag{4.2}$$

where c' is the fluctuating component of the concentration.

Our finite element discretizations of the source inversion problem require this equation to be in weak form. The weak form of equation (4.1) is given by

$$\begin{aligned} \int_\Omega \{(\nabla r) \cdot [k\nabla c - c\mathbf{U} + \mathcal{J}] - rs\}\, dA \\ - \int_{\partial\Omega} \{r(k\nabla c - c\mathbf{U} + \mathcal{J}) \cdot \mathbf{n}\}\, dl = 0, \end{aligned} \tag{4.3}$$

where Ω is the domain and $\mathbf{n}$ is the outward facing unit normal to $\partial\Omega$ and r is a test function. This equation is simplified as we account for behavior at specific types of boundaries. For our purposes, boundaries consist of walls, inlets, i.e., regions where air enters the building, and outlets which are regions where air and the toxin may exit the building. These are denoted by $\partial\Omega_W$, $\partial\Omega_I$, and $\partial\Omega_O$, respectively. In each of these areas, the turbulence term, $\mathcal{J}$, is assumed to be negligible. Since we expect that the toxin is unable to permeate walls, a no-mass-transport condition is applied at $\partial\Omega_W$, i.e.

$$[(c\mathbf{U} - k\nabla c) \cdot \mathbf{n}]_{\partial\Omega_W} = 0. \tag{4.4}$$

Notice that since we know $[\mathbf{U} \cdot \mathbf{n}]_{\partial\Omega_W} = 0$, (4.4) implies that both the convective term and the diffusive term are zero at walls. At inlets, we assume that the incoming concentration of the toxin, α, may be specified, i.e. $[(c\mathbf{U} - k\nabla c) \cdot \mathbf{n}]_{\partial\Omega_I} = -\alpha\mathbf{U} \cdot \mathbf{n}$. For now, we assume the toxin cannot enter the building through the inlet, i.e., $\alpha = 0$. Lastly, we assume that outlets have the property that

$$[(\nabla c) \cdot \mathbf{n} + \beta c]_{\partial\Omega_O} = 0, \tag{4.5}$$

where $\beta > 0$ is a loss coefficient. If the actual boundary were at infinity then we would have $[(\nabla c) \cdot \mathbf{n}]_{\partial\Omega_O} = 0$. Thus, for this work we set $\beta = 0$ to approximate our finite boundary. By incorporating these boundary conditions into (4.3), we get

$$\int_\Omega \{(\nabla r) \cdot [k\nabla c - c\mathbf{U} + \mathcal{J}] - rs\}\, dA \qquad (4.6)$$
$$- \int_{\partial\Omega_I} r\alpha \mathbf{U} \cdot \mathbf{n}\, dl + \int_{\partial\Omega_O} rc(\mathbf{U} \cdot \mathbf{n} + k\beta)\, dl = 0.$$

4.2 The Source Inversion Problem

Now we can begin to formulate the source inversion problem for the steady state case. We assume that we already have toxin concentration readings from sensors placed at fixed locations throughout the building; how we get this information for our numerical results is discussed in Chapter 6. Let c_i^* be the concentration reading from the ith sensor and $c(\mathbf{x_i})$ be the concentration computed at that location by solving (4.1). Then we want to determine the source location that results in the least discrepancy between the computed concentration distribution satisfying (4.1) and the sensor data. Thus, we define our preliminary objective function by

$$f_p(s, c) = \sum_{i=1}^{N_S} (c(\mathbf{x_i}) - c_i^*)^2, \qquad (4.7)$$

and write the optimization problem as

$$\begin{aligned} &\underset{s,\, c}{\text{minimize}} \quad f_p(s, c) \\ &\text{subject to (4.6)}, \end{aligned} \qquad (4.8)$$

where (4.6) is the weak form of the equation for the toxin transport model. Here, N_S is the number of sensor locations. While large values of N_S result in more accurate solutions [1], due to practical considerations we want N_S to be relatively small.

Notice that the the solution to (4.8) is not necessarily unique. In fact, using this formulation of the problem we can fit the concentration field, $c(\mathbf{x})$, exactly to

the sensor readings but this may result in values of $s(\mathbf{x})$ that are highly oscillatory and therefore unrealistic for our problem. We expect the source, $s(\mathbf{x})$, to be mostly zero with smooth transitions into peaks at the source locations. As a result, to prevent extraneous oscillations in the source, a regularization term that minimizes the square of the gradient of $s(\mathbf{x})$ is added to (4.7). This approach is known as Tikhonov regularization. Thus, our modified objective function is

$$f(s,c) = f_p(s,c) + \kappa \int_\Omega (\nabla s)^2, \tag{4.9}$$

where $\kappa > 0$ is a constant and our optimization problem is

$$\begin{aligned} &\underset{s,\,c}{\text{minimize}} \quad f(s,c) \\ &\text{subject to (4.6)}, \end{aligned} \tag{4.10}$$

where (4.6) is the weak form of the equation for the toxin transport model. While other types of regularization exist, as discussed in [1], applying Tikhonov regularization has the benefit of resulting in a convex quadratic objective function when (4.10) is discretized, which is easier to solve.

To apply the methods discussed in Chapter 3 it is necessary to use finite element approximations to both the objective function and the constraints in (4.10). For this work we use a software package called Sundance to construct finite element discretizations. Sundance, produced by Sandia National Laboratories, is designed for specifying, building and applying finite element approximations to general partial differential equations. Given a PDE, its associated boundary conditions, and a mesh that may be provided by the user, Sundance constructs the associated finite element mass matrix and right-hand side. This system may then either be solved via a wide range of solvers developed by Sandia National Laboratories, as is the case with the equality constrained problem, or given as input to an optimization package like O3D, as we discuss later in this chapter. Details concerning the use and implementation of

Sundance will be forthcoming in [1], [5], [8] and [9].

It is important to notice two properties of this formulation of the problem. First, the finite element approximation to (4.10) is an equality constrained convex quadratic programming problem which, as we discussed in Section 3.2, is relatively easy to solve. Second, it is necessary to solve for both the concentration and the source at every point $\mathbf{x}$ where we require a solution, i.e., the problem is large. It is possible to significantly reduce the number of computed variables by assuming $s(\mathbf{x})$ has a specific form, for example a Gaussian centered at each source location. However, there are two problems with this. First, we would need to know beforehand how many sources existed in the building and second, every model for the source that we have considered is either discontinuous or results in nonlinear constraints, or an objective function that is not quadratic, which adds computational difficulties.

While solutions to (4.10) often give adequate results, it is common to find that $s(\mathbf{x})$ contains negative values. Since this is not realistic and produces more non-physical oscillations, we can also require that $s(\mathbf{x})$ is non-negative. Thus, our problem is redefined as

$$\begin{aligned} \underset{s,\,c}{\text{minimize}} \quad & f(s,c) \\ \text{subject to} \quad & (4.6) \\ & s \geq 0, \end{aligned} \tag{4.11}$$

where (4.6) is the weak form of the equation for the toxin transport model. If we then use Sundance to construct the finite element approximations to the equations in (4.11), we can obtain a solution to the source inversion problem by applying the optimization algorithm called O3D which we described in Chapter 3. As discussed in Chapter 3, in order to solve inequality constrained problems with O3D it is necessary to incorporate equality constraints into the objective function via a quadratic penalty term. This quadratic penalty formulation of the source inversion problem allows us to control the accuracy with which the equality constraints are satisfied. Since we

are using a finite element approximation to the equality constraints, it does not make sense to require (4.6) to be satisfied with high accuracy. Also, notice that the strictly feasible starting point required by Algorithm 3.1 can be obtained by solving the equality constrained problem (4.10) and altering the solution so that $s(\mathbf{x})$ is strictly positive.

Chapter 5

Mesh Strategies

In Chapter 2, we mentioned that due to the large Reynolds number, it is necessary to solve for the flow field on a fine mesh. Since the flow field may be calculated beforehand, computation time is not of great concern when solving the flow model. However, time is of great importance when solving the source-inversion problem. Thus, after the flow field is calculated, we would like to reduce the number of triangles in the mesh, and thereby the number of variables in the source-inversion problem without introducing a significant loss in accuracy. Here, we only consider two different types of meshes, uniform and adaptive, which we describe in the sections below.

In order to generate these meshes, we use BAMG, which stands for Bidimensional Anisotropic Mesh Generator and is available through Institut National de Recherche en Informatique et en Automatique (INRIA). BAMG produces a mesh based on a given two-dimensional geometry. More importantly for this work, BAMG can alter a pre-existing mesh based on certain properties of a given solution on that mesh. Specifically, given a field, the mesh on which it was computed and a metric, BAMG calculates the value of the metric, which is a 2×2 symmetric positive definite matrix M_i, at each vertex i in the mesh. M_i is then used to determine the mesh size, h_i, in

any direction $\mathbf{v}$ from vertex i via the formula

$$h_i(\mathbf{v}) = \frac{||\mathbf{v}||}{\sqrt{\mathbf{v}^{\mathrm{T}} M_i \mathbf{v}}}. \tag{5.1}$$

Now, based on the values for h_i and user given requirements such as the maximum and minimum edge length, BAMG constructs a new mesh. Finally, the flow field information is transferred to the new mesh via P_1 Lagrange interpolation. In the sections below, we give details concerning the mesh strategies that we use and some of the features of BAMG that are required to produce these meshes. For further information concerning BAMG we refer the reader to [7].

5.1 Uniform Meshes

A uniform mesh, for our purposes, refers to a mesh in which all edge lengths are approximately the same. In BAMG, this is accomplished by specifying M_i to be a positive scalar multiple of the 2×2 identity matrix at every vertex in the mesh, i.e., $M_i = M = aI$ for all i, where $a > 0$ is the same for every vertex in the mesh. Thus, we have

$$h = \frac{||\mathbf{v}||}{\sqrt{\mathbf{v}^{\mathrm{T}} M \mathbf{v}}} = \frac{1}{\sqrt{a}}\ , \tag{5.2}$$

for every unit vector $\mathbf{v}$. Now we define the edge length by

$$e = e_{\max} = e_{\min}, \tag{5.3}$$

where $e_{\max}$ and $e_{\min}$ are, respectively, the maximum and minimum edge lengths for every edge in the mesh. Therefore, if we let

$$a = \frac{1}{e^2} \tag{5.4}$$

then for each vertex in the mesh, $h = e$ independent of the direction vector $\mathbf{v}$. As a result, when BAMG constructs a uniform mesh, it attempts to have each vertex a

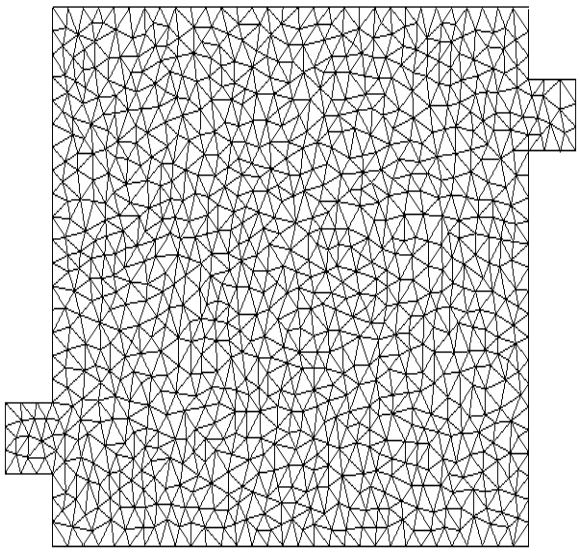

Figure 5.1: Uniform mesh with $h = 0.64$ and 1826 triangles.

distance of exactly e from all adjacent vertices. Unfortunately, instances arise in which BAMG must adjust certain edge lengths in order to create the mesh and consequently the edge lengths are not exactly the same, but they are close. An example of a uniform mesh generated by BAMG is shown in Figure 5.1.

5.2 Adaptive Meshes

It is important to remember that after a mesh is generated, BAMG projects the flow field onto this new mesh via P_1 Lagrange interpolation, i.e., linear interpolation. Thus, we are assuming that in an area defined by the elements adjacent to any given vertex, the flow field can be approximated by a linear function. Unfortunately, as we coarsen the mesh, this may not always be true. Thus, a more intelligent choice for constructing a coarser mesh is to have the mesh size in a given direction from vertex i depend upon how quickly the flow field is changing in that direction, i.e., the curvature of the flow field in that direction. We refer to meshes with this property as

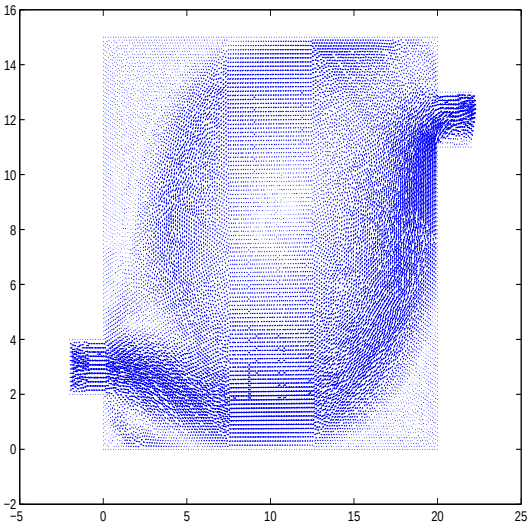

Figure 5.2: Flow field used to construct the adaptive mesh.

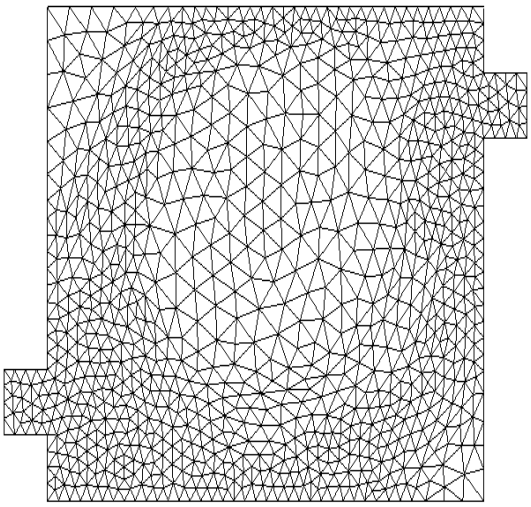

Figure 5.3: Adaptive mesh with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

adaptive meshes; see Figures 5.2–5.3 for an example.

Since at each vertex in the flow field we have both an x-component and a y-component, it is necessary to calculate two metric values for each vertex and then combine them later. For the meshes we use, the metric for the x-component of the flow field is defined by

$$M_i^x = \delta \frac{|\mathcal{H}_x|}{\sup u_x - \inf u_x}, \tag{5.5}$$

where $\delta > 0$ is constant, u_x is the x-component of the flow field, and $\mathcal{H}_x$ is the Hessian of u_x. Here, $|\mathcal{H}_x|$ is defined by

$$|\mathcal{H}_x| = P^{-1}|A|P, \tag{5.6}$$

where P is a matrix with column i containing the i^{th} eigenvector of $\mathcal{H}_x$ and $|A|$ is a diagonal matrix with the absolute value of the i^{th} eigenvalue of $\mathcal{H}_x$ in diagonal position i. The metric for the y-component of the flow field, M_i^y, is similar to (5.5). For each vertex i, the metrics M_i^x and M_i^y define ellipses by the equations

$$\mathbf{w}^{\mathrm{T}} M_i^x \mathbf{w} = 1 \tag{5.7}$$

and

$$\mathbf{w}^{\mathrm{T}} M_i^y \mathbf{w} = 1, \tag{5.8}$$

where $\mathbf{w}$ is a vector. Thus the metric, M_i, that is used to compute the mesh size is the metric that results in the largest ellipse contained in the ellipses given by (5.7) and (5.8). While using this definition only guarantees that M_i is positive semi-definite as opposed to positive definite, we did not encounter the situation where $\mathbf{v}^{\mathrm{T}} M_i \mathbf{v} = 0$.

Now using M_i we can calculate the mesh size via (5.1) and define the edge length in a given direction $\mathbf{v}$ by

$$e_v = e_{\min} + \frac{h_i(\mathbf{v})}{\max\limits_i h_i(\mathbf{v})}(e_{\max} - e_{\min}), \tag{5.9}$$

where $e_{\max}$ and $e_{\min}$ are, respectively, the maximum and minimum edge lengths for every edge in the mesh. An adaptive mesh is then created so that an edge in direction $\mathbf{v}$ from vertex i has edge length e_v.

Chapter 6

Numerical Results

To test our source inversion approach to this problem, it is necessary to have two pieces of information. First, it is necessary to have the solution to the flow model discussed in Chapter 2 and second, we need some means of generating sensor data. Both of these issues are discussed in the following sections. At the end of this chapter, we discuss some of our numerical results.

6.1 Flow Field and Geometry

The flow field that is used for this work was computed on a 15×20 square units building with one inlet and one outlet which are each 2×2 square units; see Figure 6.2. This particular flow field has the inlet velocity profile specified to be Poiseuille, i.e., parabolic with respect to the x-direction and zero in the y-direction and at the walls, with the peak velocity at the center of the inlet taken to be 4. In addition, the Reynolds number is taken to be 10000. As mentioned in Chapter 2, due to the complexity of the air flow in the types of buildings we consider, it is necessary to solve the flow model on a fine mesh. This mesh is shown in Figure 6.1.

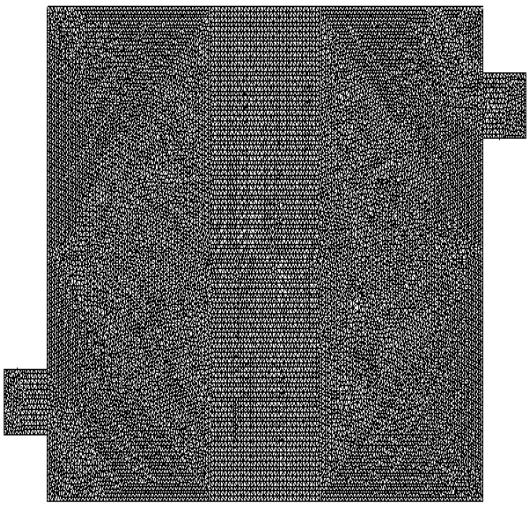

Figure 6.1: Mesh with $h = 0.15$ and $31,602$ triangles. This mesh is used to calculate the flow field.

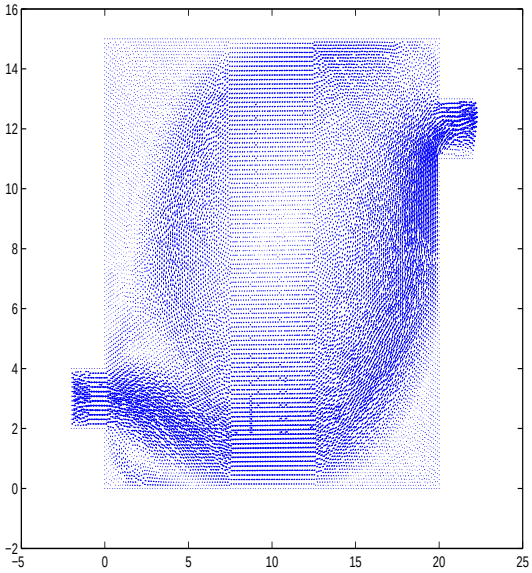

Figure 6.2: Calculated vector flow field for $Re = 10000$.

6.2 Sensor Data

Since we do not have access to actual concentration data, it is necessary to generate sensor data for our two dimensional example. In order to create sensor data we first model each source as a Gaussian. Specifically, we define $s(\mathbf{x})$ in (4.6) to be

$$s(\mathbf{x}) = \xi_i e^{-\upsilon_i(\mathbf{x}-\mathbf{x}_i)^2}, \tag{6.1}$$

where $\xi_i = 20$, $\upsilon_i = 2$ and $\mathbf{x}_i$ is the location at which the source is centered. Now, the weak form of the dispersion equation (4.6) can be solved using Sundance to obtain the concentration, $c(\mathbf{x})$. We then determine the value of $c(\mathbf{x})$ at fixed sensor locations and modify it by a random amount within 5%. To minimize accuracy loss when generating the sensor data, we solve for $c(\mathbf{x})$ using the mesh and flow field shown in Figures 6.1 and 6.2.

In order to determine an effective sensor arrangement, it is necessary to consider

how the toxin is transported throughout the building and how this information is represented by the sensor readings. Here, we will look at two different cases. First, suppose a source is located in a region of the building in which the air flow is minimal, e.g., a corner. Over time the toxin will accumulate and any sensor in that region will exhibit a high concentration reading while all other sensors return comparatively low readings, assuming only one source exists. Thus, given at least one sensor in this region, any solution would exhibit a source in the proper area of the building. Since any sensor in this location will eventually have a comparatively high concentration reading, we require only one sensor to adequately determine sources in these types of regions. For the second case, suppose we have a source in the main stream of the flow. Instead of accumulating around the source location, the toxin will be distributed throughout the building as dictated by the flow field. Thus, if the main stream of the flow contains too few sensors many different source locations could result in the same set of concentration readings. Therefore, in the main stream of the flow it is necessary to place sensors based on both the direction and the magnitude of the flow. As a result, we can reconstruct the source and concentration better by using an irregular placement of sensors. A comparison of results using regular versus irregular sensor placement will be forthcoming in [5].

For the results in this work, 30 sensor locations were selected by hand; see Figure 6.3. While we do not claim that this sensor arrangement is optimal for this geometry and flow field, we have found that using these sensor locations we were able to locate reasonable test sources with acceptable accuracy.

6.3 Discussion of the Numerical Results

Since it is not necessary to show all the numerical results in order to understand the issues we address, only some of the results are presented in this section. The

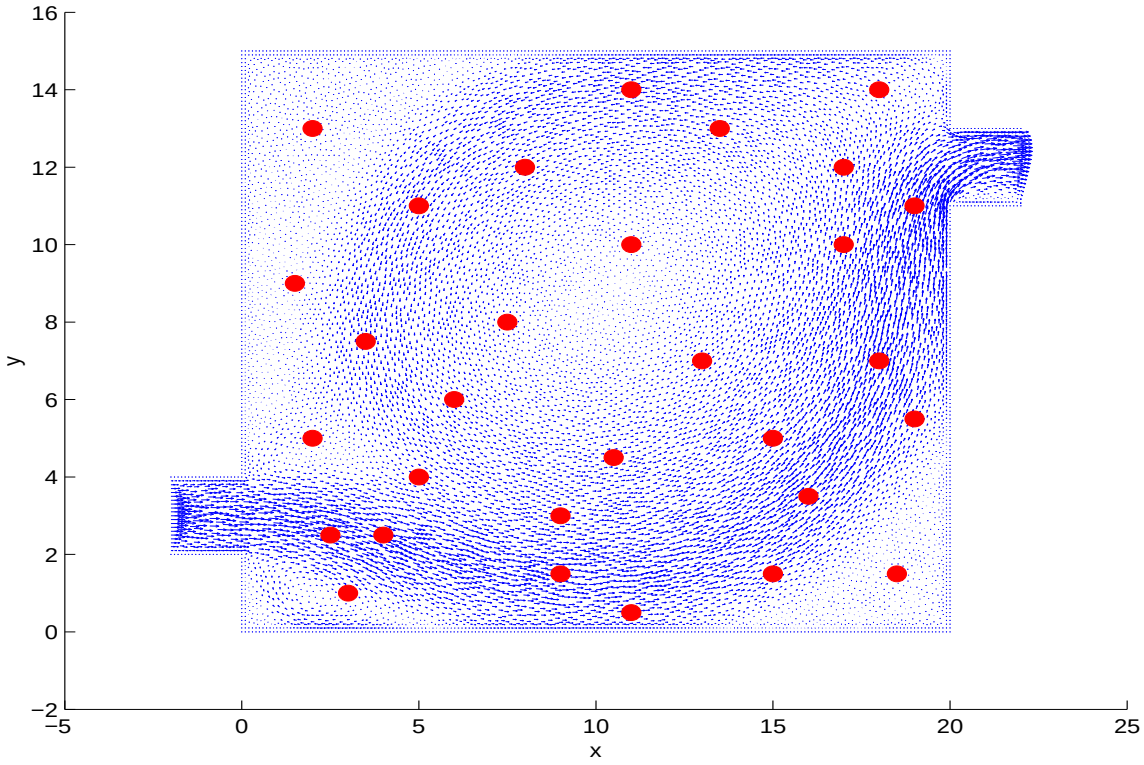

Figure 6.3: Flow field with 30 hand selected sensor locations.

majority of the numerical results can be found in the Appendix. The meshes used to generate these results can also be found in the Appendix.

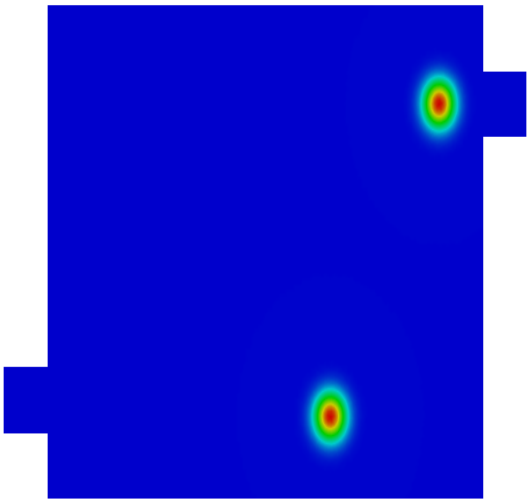

Figure 6.4: Actual source.

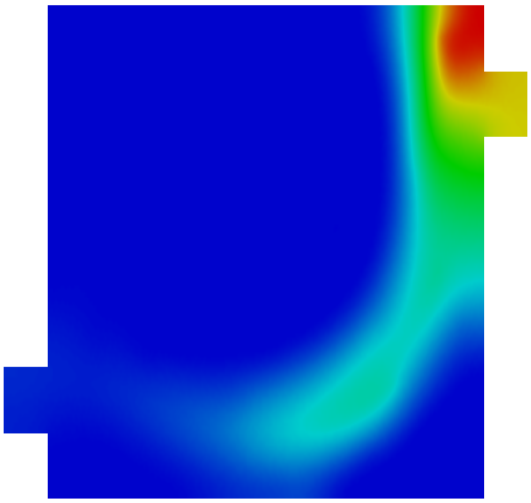

Figure 6.5: Computed source with $h = 0.15$ and $31,602$ triangles.

The first issue that we address is the accuracy of our computed source field. Even when the source inversion problem is solved on the flow field mesh shown in Figure 6.1, the sensor arrangement we use does not locate certain sources well, as shown in Figures 6.4–6.5, or at all, as shown in Figures 6.6–6.7. In the case of Figure 6.6–6.7, nearly all of the substance released in the rectangular region defined by $[18, 20] \times [7, 11]$ leaves the building through the outlet; see Figure 6.2. As a result, the toxin released by a source in this region is only detected by one sensor and thus we expect these types of sources to be difficult to locate. In addition, since nearly all of the toxin from this source is leaving the building anyway, we assume that this source is not as dangerous to the people in the building. Thus, we do not consider this source location to be as important as other possible source locations and therefore this result is adequate for the time being. If, in the future, we decide that this is not an acceptable result, we

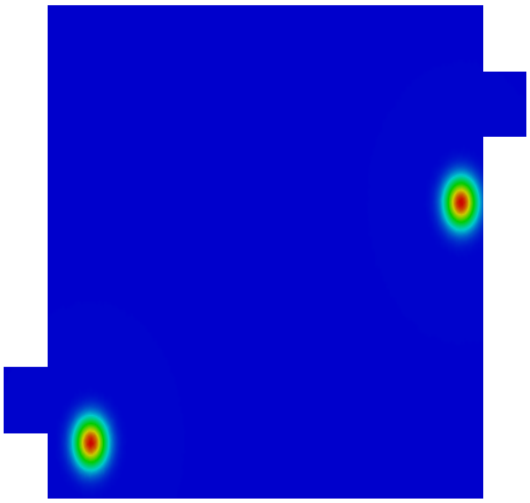

Figure 6.6: Actual source.

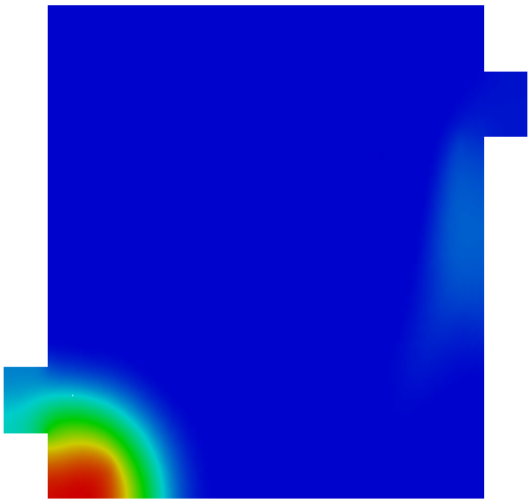

Figure 6.7: Computed source with $h = 0.15$ and $31,602$ triangles.

believe that placing sensors in the outlet will remedy this situation. Note that results in the appendix show that if there exists only one source in the building and it is located in the region $[18, 20] \times [7, 11]$, the computed solution clearly indicates a source in the appropriate area. For the situation in Figures 6.4–6.5, in which the computed source is smeared, we believe that the solution is still adequate since an individual with knowledge of the flow field still would be able to predict the region of the source location correctly. In addition, when we expand this problem to account for time dependence, we will have sensor readings for each time increment. Consequently, we have significantly more information about how the substance is spreading and thus we expect our time dependent solutions will have greater accuracy.

We solved the source inversion problem using a variety of different meshes and source locations. Some of these results for the source are shown in Figures 6.8–6.17. The results for the concentration are not presented for two reasons. First, the reconstruction of the concentration field is significantly better than that of the source field, which is an indication that this problem is ill-conditioned and second, accuracy

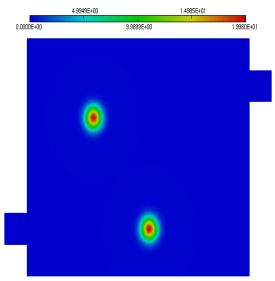

Figure 6.8: Actual source.

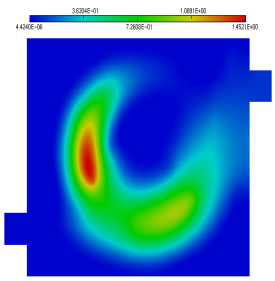

Figure 6.9: Computed source with $h = 0.15$ and 31,602 triangles.

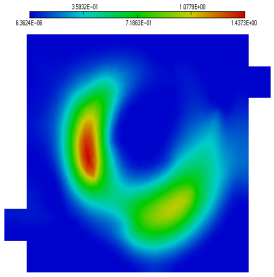

Figure 6.10: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

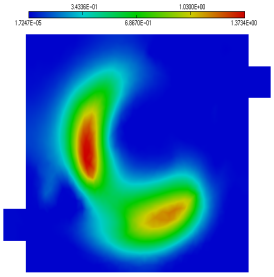

Figure 6.11: Computed source with $h = 0.52$, and 2704 triangles.

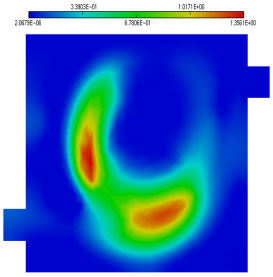

Figure 6.12: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

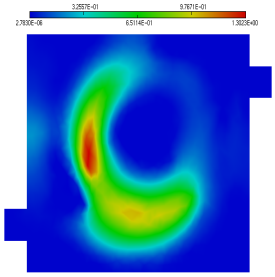

Figure 6.13: Computed source with $h = 0.64$, and 1826 triangles.

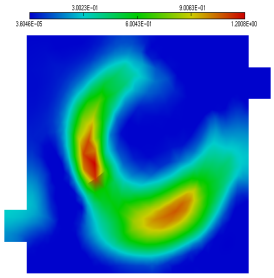

Figure 6.14: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

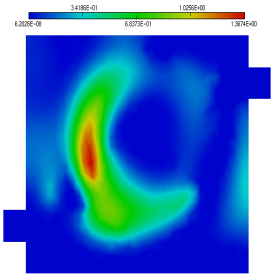

Figure 6.15: Computed source with $h = 0.72$, and 1488 triangles.

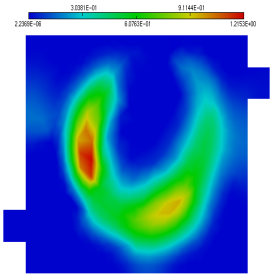

Figure 6.16: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

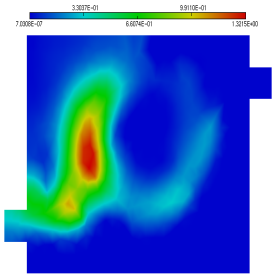

Figure 6.17: Computed source with $h = 0.8$, and 1196 triangles.

in the source field is of greater concern for the scenario that we are considering. Here, the actual value of the source is given in Figure 6.8 and the result from the source inversion problem using the flow field mesh is given in Figure 6.9. Figures 6.10–6.11 show the results on meshes with progressively fewer triangles. Notice that the computed source using the flow field mesh is nearly indistinguishable from the result using a mesh with over 90% fewer triangles, as in Figures 6.10–6.17. Since for the time-dependent problem we will be solving for the values of both the source and concentration at every time increment, the impact of this sort of reduction will be significant. Similarly, the ability to get this level of accuracy in solutions on coarse meshes will be a great benefit when the problem is expanded to account for three dimensions.

These results also show that if we use adaptive meshes, we can reduce the number of triangles a little bit more without much more loss in accuracy. Notice that for the results calculated on meshes containing approximately 2700 triangles, there is very little difference between the reconstructed source using the adaptive mesh and that found using the uniform mesh, as in Figures 6.10–6.11. However, if we continue to decrease the number of triangles, it becomes apparent that it may be beneficial to choose an adaptive meshing strategy over a uniform one. Figure 6.15 shows that once we decrease the number of triangles to approximately 1500, the reconstructed source on a uniform mesh no longer clearly indicates one of the sources. At the bottom source location, instead of the strong peak that we see in Figure 6.16, Figure 6.17 shows comparatively low source values that could easily be misinterpreted as smearing from the top source.

Finally, many different adaptive meshes have the same number of triangles but not all of them give good results when applied to this problem. The choice of minimum and maximum edge lengths affects the accuracy of the solution to the source inversion problem. After these meshes are generated, BAMG interpolates the value of the flow

field at each point in this new mesh; see Chapter 5. As a result, there are errors in the flow values used to solve the source inversion problem. If the edge length is too big, then the error in the flow field can cause extra sources or excessive smearing to appear in the reconstructed source. Note that this is also the reason that adaptive meshes provide better results than uniform meshes as the number of triangles decreases. Please see the Appendix for further results.

Chapter 7

Future Work

The results in this thesis show that by using coarse meshes we can not only obtain adequate results but, in general, we can achieve a significant reduction in the number of triangles in the mesh before a loss in accuracy is noticeable. Nonetheless, there is quite a bit of work left. While much of the work done with the steady-state case may be applied to the time-dependent case, the time-dependent problem is overall much more complicated. For the time-dependent case, we solve for both the concentration and the source fields at every time increment. Thus, we have significantly more unknowns but we also have more information about how the toxin is permeating the building since sensor data is also retrieved at every time increment. In addition, while we know when the toxin first reaches a sensor, it is not known how long the source was present prior to the first sensor reading. The best strategy for determining the time that the source first appeared is not clear. In addition, for the time-dependent case, we will need to consider the effect of multiple sources appearing at different time steps and moving sources.

The sensor locations that we used for the results presented here were hand-selected. While the results obtained using this sensor arrangement were adequate, we

would like to find some concrete means of determining an optimal sensor arrangement. This problem is hard since it is not clear how to formulate an appropriate objective function. One approach is to minimize the difference between the actual source field and the computed source field. While there are infinitely many different possibilities for the source field, we believe it will be necessary to solve this problem using a finite set of source fields. Unfortunately, different types of toxins dictate different attack strategies so the best way to choose a set of source fields is not obvious.

Many of the choices that were made in developing the current problem formulation reflect the desire to preserve the quadratic program structure. In particular, this is a benefit to using standard Tikhonov regularization. Unfortunately the results show that this tends to smear the source. As a result, it may be beneficial to consider other types of regularization; see [1].

Appendix A

Meshes for Numerical Results

All the mesh images presented in this thesis were generated using MEDIT. Information concerning this graphics software can be found in [6].

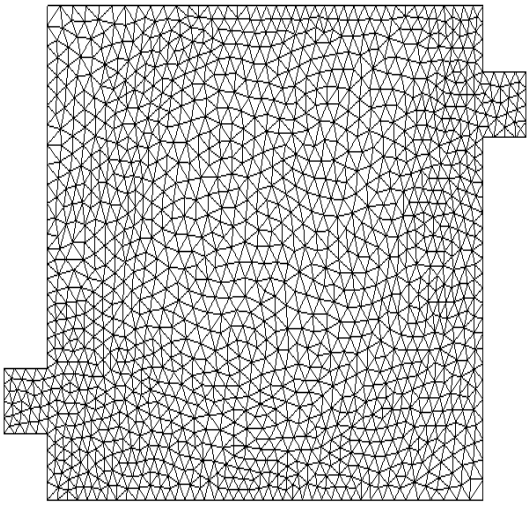

Figure A.1: Mesh with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

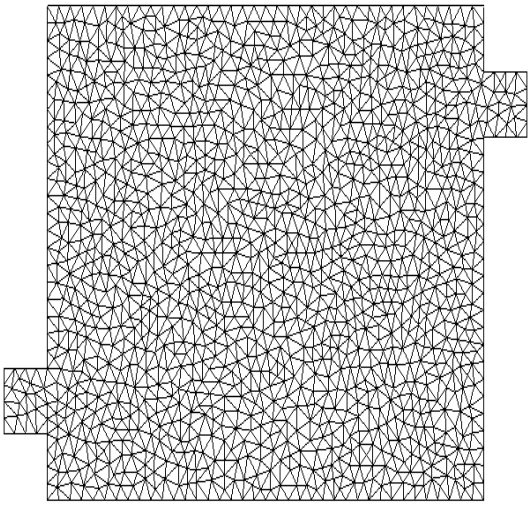

Figure A.2: Mesh with $h = 0.52$ and 2704 triangles.

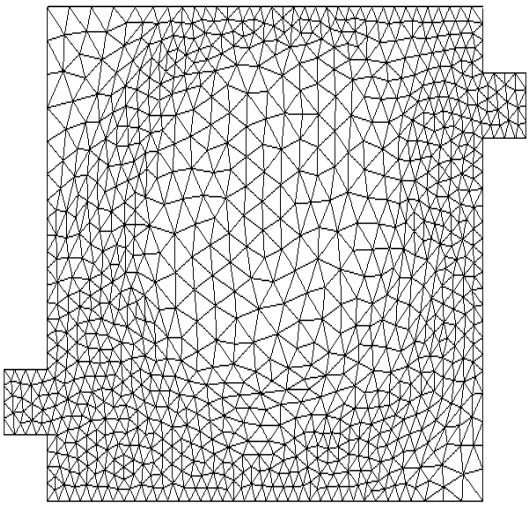

Figure A.3: Mesh with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

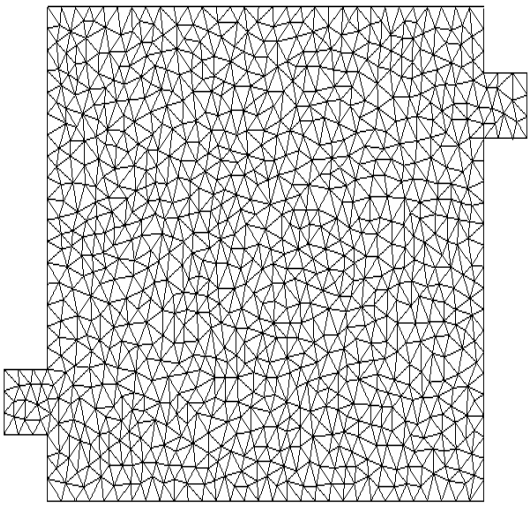

Figure A.4: Mesh with $h = 0.64$ and 1826 triangles.

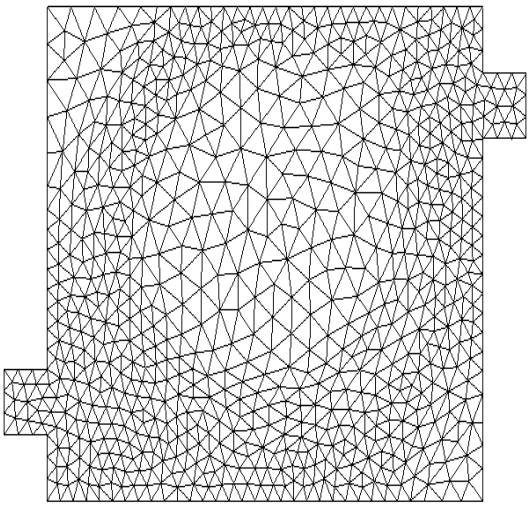

Figure A.5: Mesh with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

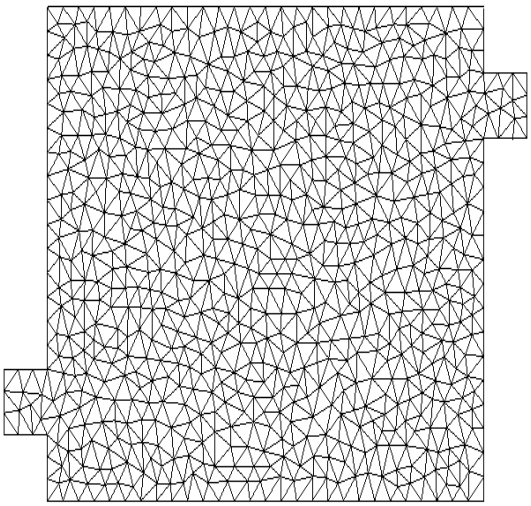

Figure A.6: Mesh with $h = 0.72$ and 1488 triangles.

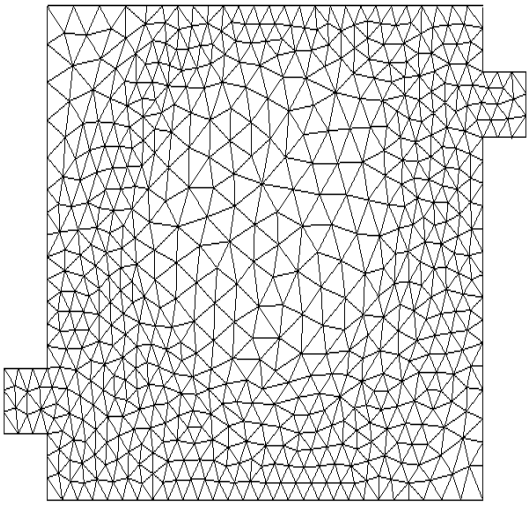

Figure A.7: Mesh with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

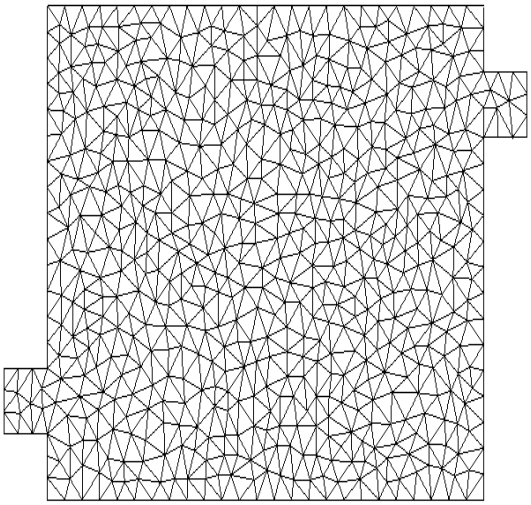

Figure A.8: Mesh with $h = 0.8$ and 1196 triangles.

Appendix B

More Numerical Results

All the source field images presented in this thesis were generated using MEDIT. Information concerning this graphics software can be found in [6].

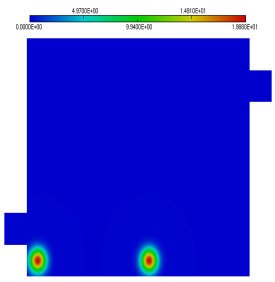

Figure B.1: Actual source.

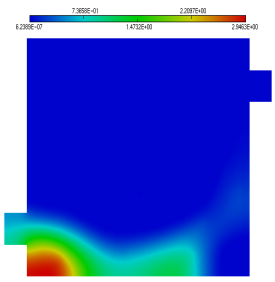

Figure B.2: Computed source with $h = 0.15$ and 31,602 triangles.

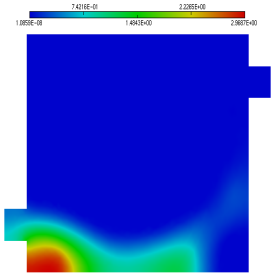

Figure B.3: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

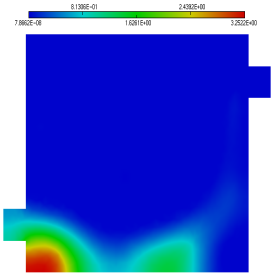

Figure B.4: Computed source with $h = 0.52$, and 2704 triangles.

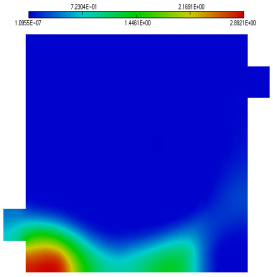

Figure B.5: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

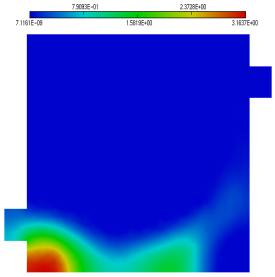

Figure B.6: Computed source with $h = 0.64$, and 1826 triangles.

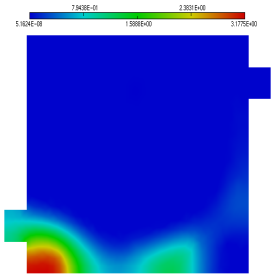

Figure B.7: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

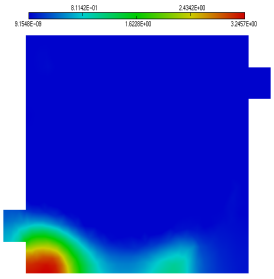

Figure B.8: Computed source with $h = 0.72$, and 1488 triangles.

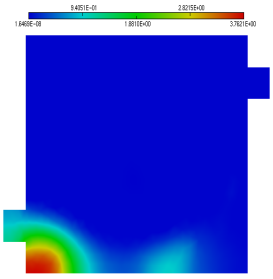

Figure B.9: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

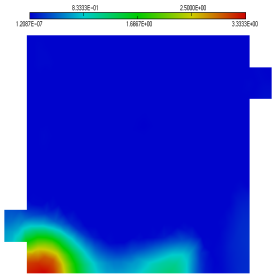

Figure B.10: Computed source with $h = 0.8$, and 1196 triangles.

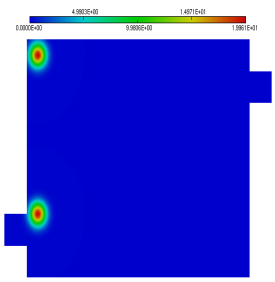

Figure B.11: Actual source.

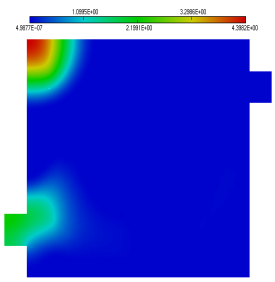

Figure B.12: Computed source with $h = 0.15$ and $31,602$ triangles.

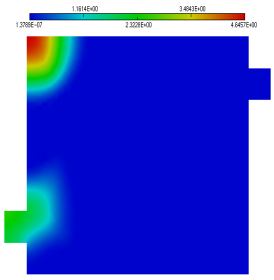

Figure B.13: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

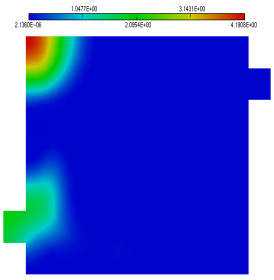

Figure B.14: Computed source with $h = 0.52$, and 2704 triangles.

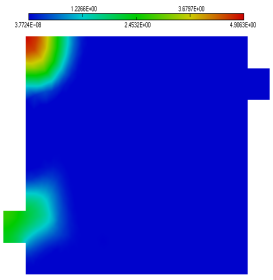

Figure B.15: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

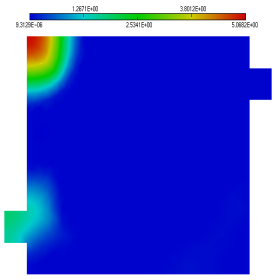

Figure B.16: Computed source with $h = 0.64$, and 1826 triangles.

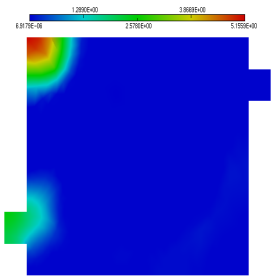

Figure B.17: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

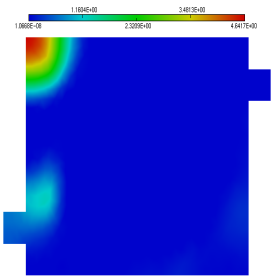

Figure B.18: Computed source with $h = 0.72$, and 1488 triangles.

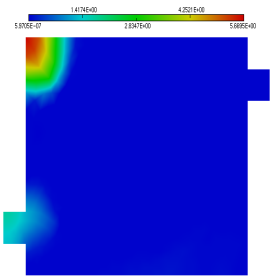

Figure B.19: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

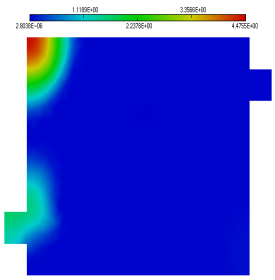

Figure B.20: Computed source with $h = 0.8$, and 1196 triangles.

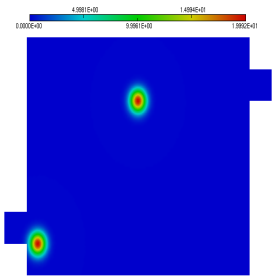

Figure B.21: Actual source.

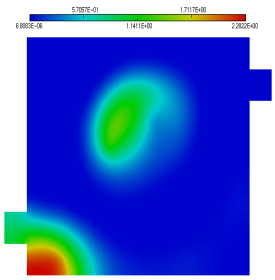

Figure B.22: Computed source with $h = 0.15$ and $31,602$ triangles.

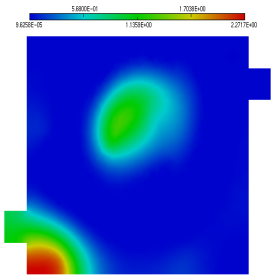

Figure B.23: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

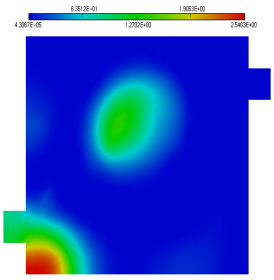

Figure B.24: Computed source with $h = 0.52$, and 2704 triangles.

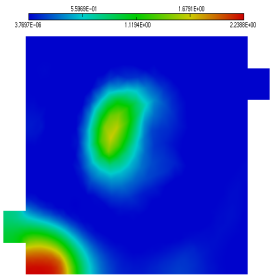

Figure B.25: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

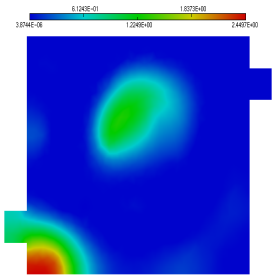

Figure B.26: Computed source with $h = 0.64$, and 1826 triangles.

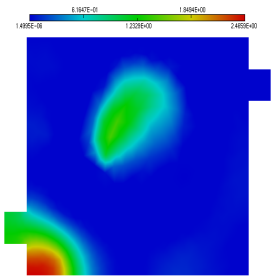

Figure B.27: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

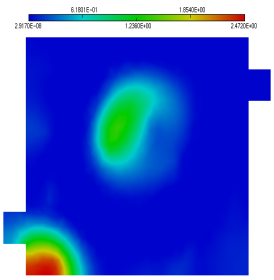

Figure B.28: Computed source with $h = 0.72$, and 1488 triangles.

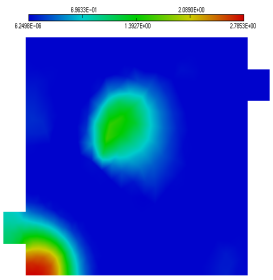

Figure B.29: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

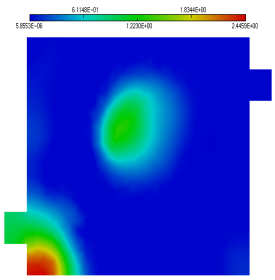

Figure B.30: Computed source with $h = 0.8$, and 1196 triangles.

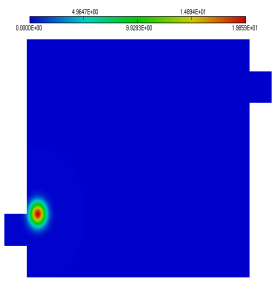

Figure B.31: Actual source.

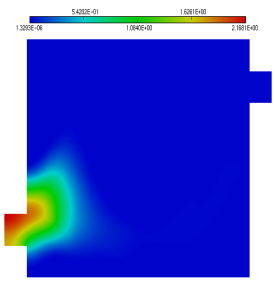

Figure B.32: Computed source with $h = 0.15$ and $31,602$ triangles.

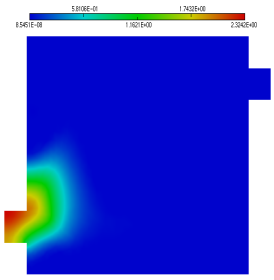

Figure B.33: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

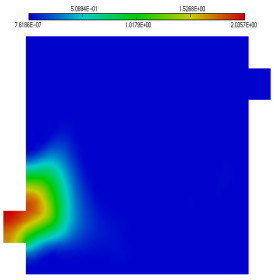

Figure B.34: Computed source with $h = 0.52$, and 2704 triangles.

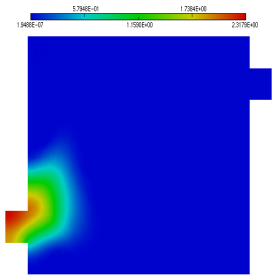

Figure B.35: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

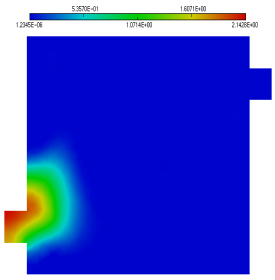

Figure B.36: Computed source with $h = 0.64$, and 1826 triangles.

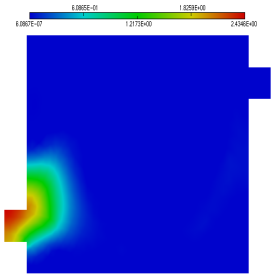

Figure B.37: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

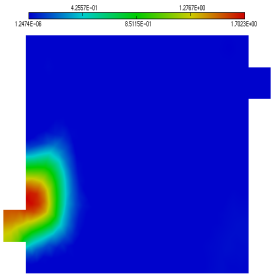

Figure B.38: Computed source with $h = 0.72$, and 1488 triangles.

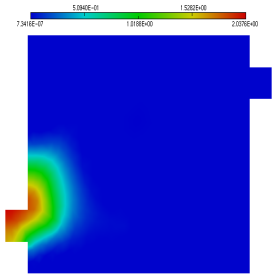

Figure B.39: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

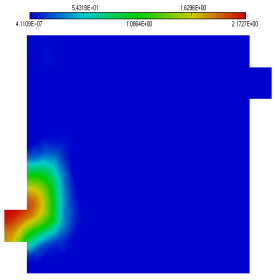

Figure B.40: Computed source with $h = 0.8$, and 1196 triangles.

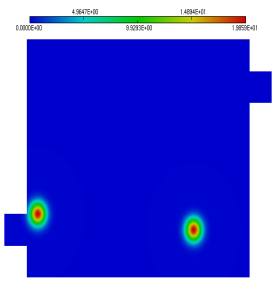

Figure B.41: Actual source.

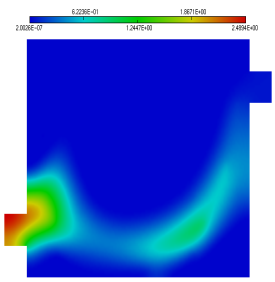

Figure B.42: Computed source with $h = 0.15$ and $31,602$ triangles.

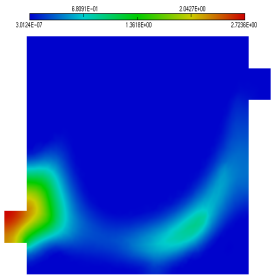

Figure B.43: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

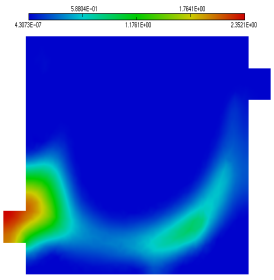

Figure B.44: Computed source with $h = 0.52$, and 2704 triangles.

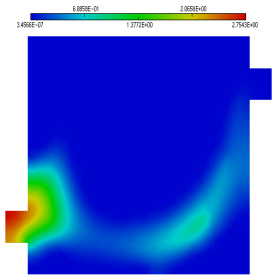

Figure B.45: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

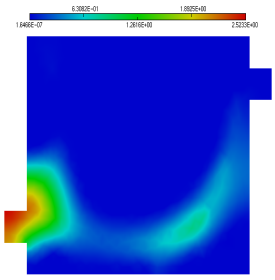

Figure B.46: Computed source with $h = 0.64$, and 1826 triangles.

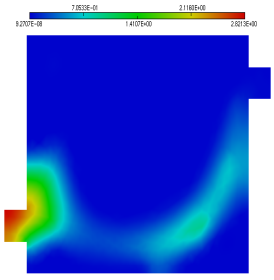

Figure B.47: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

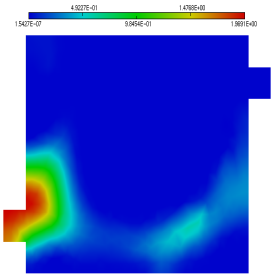

Figure B.48: Computed source with $h = 0.72$, and 1488 triangles.

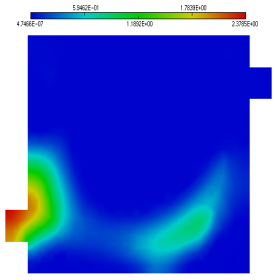

Figure B.49: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

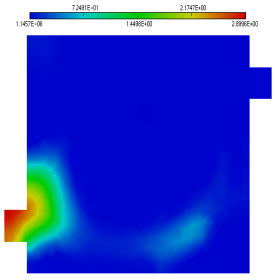

Figure B.50: Computed source with $h = 0.8$, and 1196 triangles.

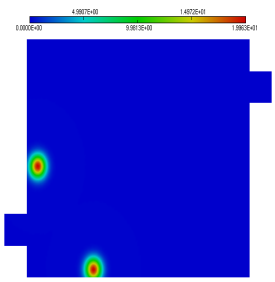

Figure B.51: Actual source.

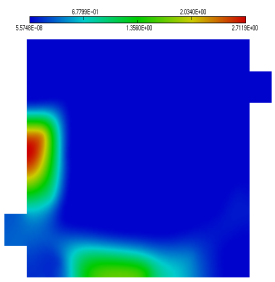

Figure B.52: Computed source with $h = 0.15$ and $31,602$ triangles.

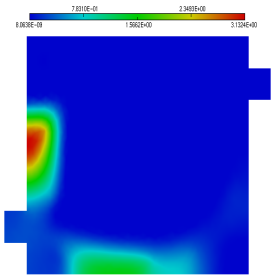

Figure B.53: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

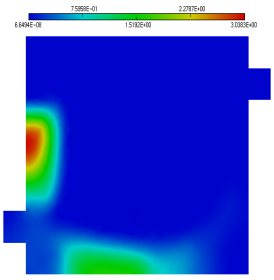

Figure B.54: Computed source with $h = 0.52$, and 2704 triangles.

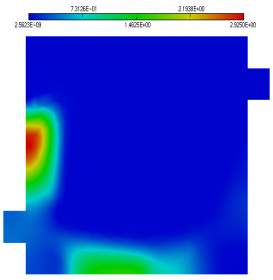

Figure B.55: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

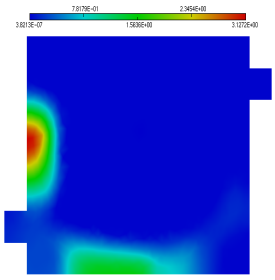

Figure B.56: Computed source with $h = 0.64$, and 1826 triangles.

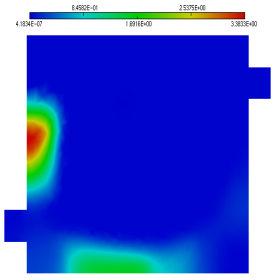

Figure B.57: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

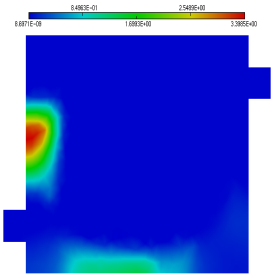

Figure B.58: Computed source with $h = 0.72$, and 1488 triangles.

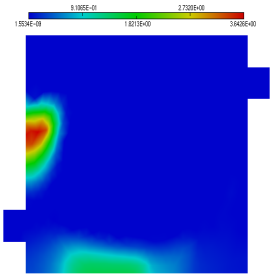

Figure B.59: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

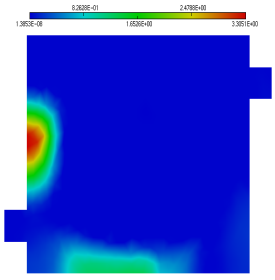

Figure B.60: Computed source with $h = 0.8$, and 1196 triangles.

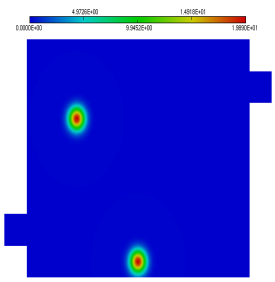

Figure B.61: Actual source.

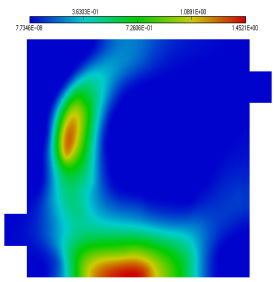

Figure B.62: Computed source with $h = 0.15$ and $31,602$ triangles.

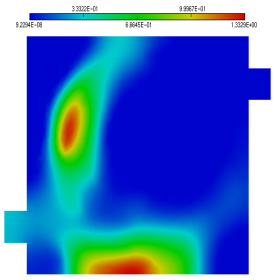

Figure B.63: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

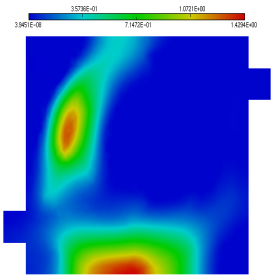

Figure B.64: Computed source with $h = 0.52$, and 2704 triangles.

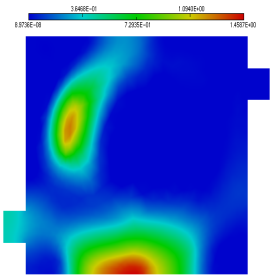

Figure B.65: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

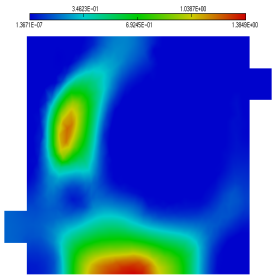

Figure B.66: Computed source with $h = 0.64$, and 1826 triangles.

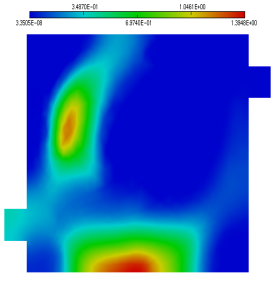

Figure B.67: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

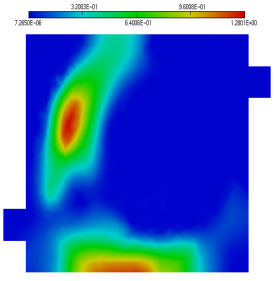

Figure B.68: Computed source with $h = 0.72$, and 1488 triangles.

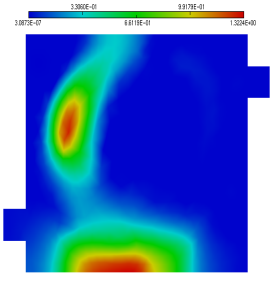

Figure B.69: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

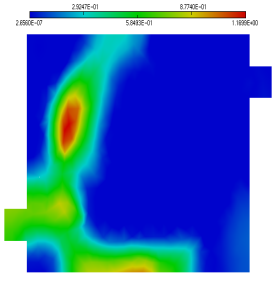

Figure B.70: Computed source with $h = 0.8$, and 1196 triangles.

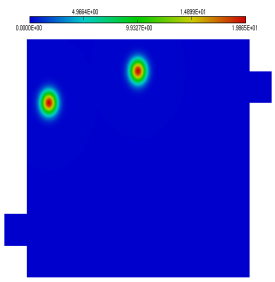

Figure B.71: Actual source.

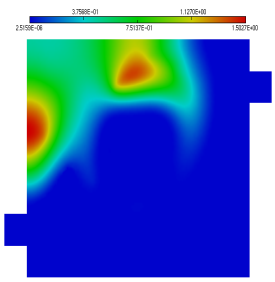

Figure B.72: Computed source with $h = 0.15$ and $31,602$ triangles.

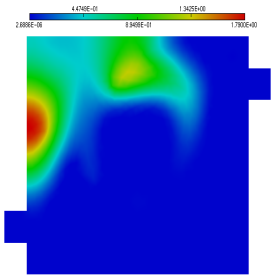

Figure B.73: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

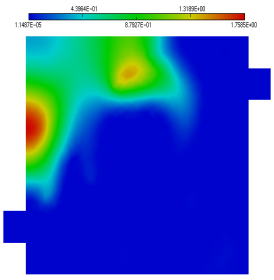

Figure B.74: Computed source with $h = 0.52$, and 2704 triangles.

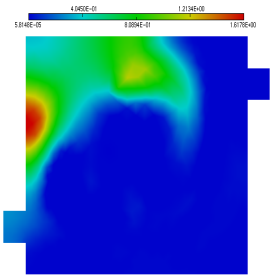

Figure B.75: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

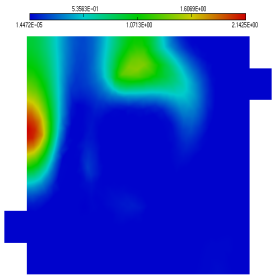

Figure B.76: Computed source with $h = 0.64$, and 1826 triangles.

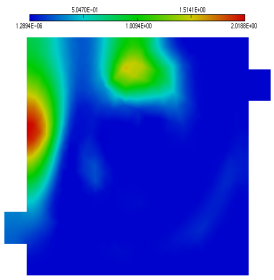

Figure B.77: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

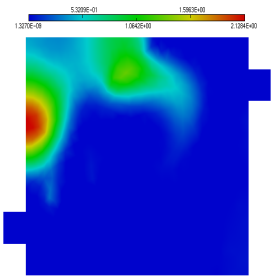

Figure B.78: Computed source with $h = 0.72$, and 1488 triangles.

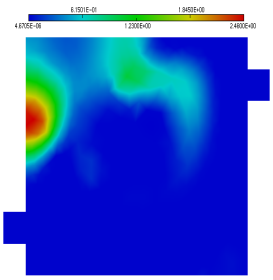

Figure B.79: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

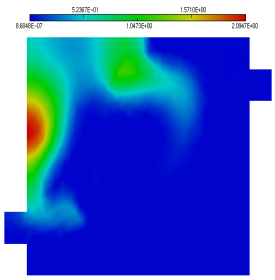

Figure B.80: Computed source with $h = 0.8$, and 1196 triangles.

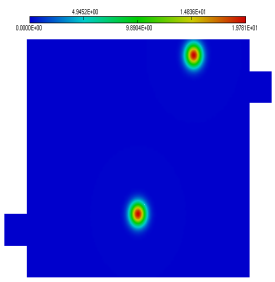

Figure B.81: Actual source.

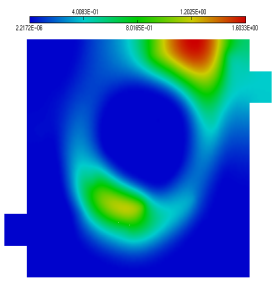

Figure B.82: Computed source with $h = 0.15$ and $31,602$ triangles.

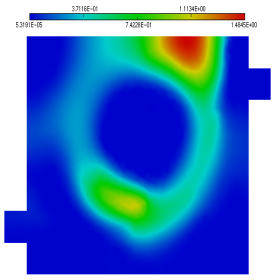

Figure B.83: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

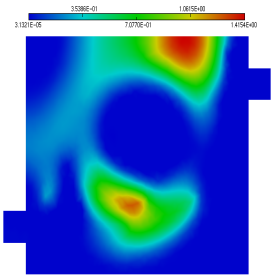

Figure B.84: Computed source with $h = 0.52$, and 2704 triangles.

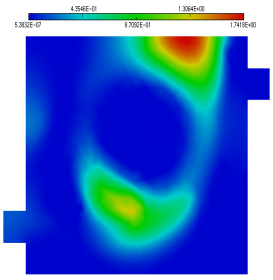

Figure B.85: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

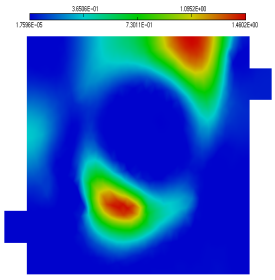

Figure B.86: Computed source with $h = 0.64$, and 1826 triangles.

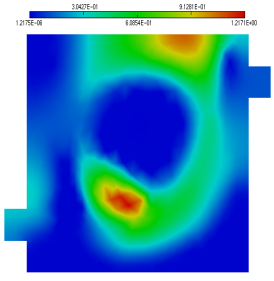

Figure B.87: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

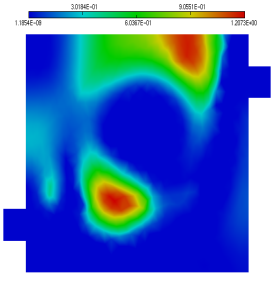

Figure B.88: Computed source with $h = 0.72$, and 1488 triangles.

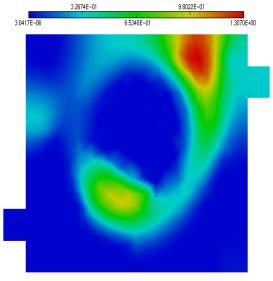

Figure B.89: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

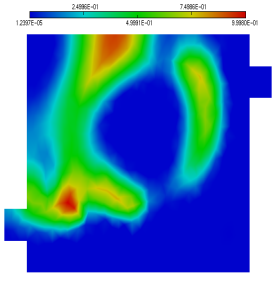

Figure B.90: Computed source with $h = 0.8$, and 1196 triangles.

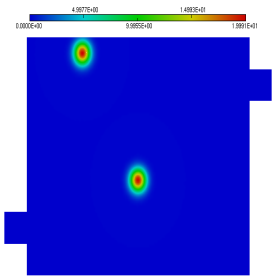

Figure B.91: Actual source.

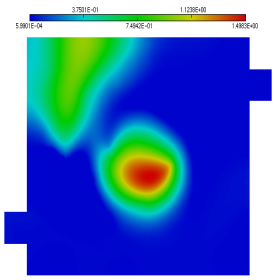

Figure B.92: Computed source with $h = 0.15$ and $31,602$ triangles.

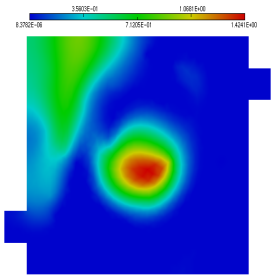

Figure B.93: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

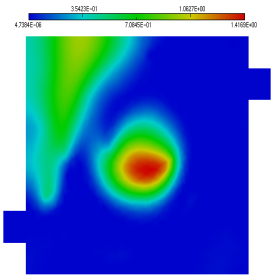

Figure B.94: Computed source with $h = 0.52$, and 2704 triangles.

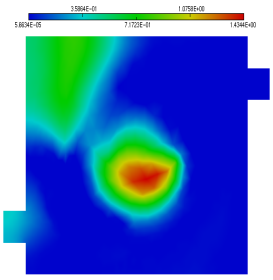

Figure B.95: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

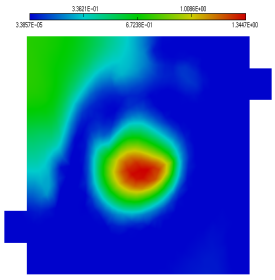

Figure B.96: Computed source with $h = 0.64$, and 1826 triangles.

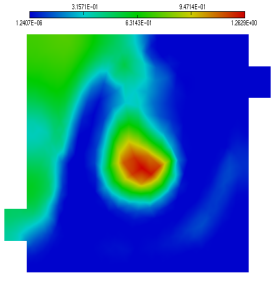

Figure B.97: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

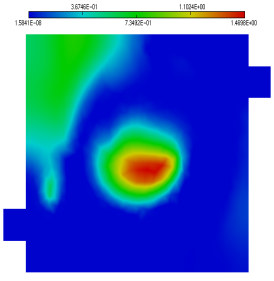

Figure B.98: Computed source with $h = 0.72$, and 1488 triangles.

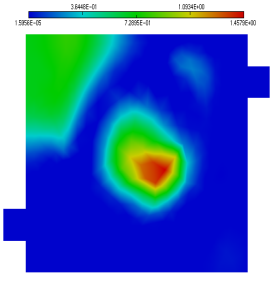

Figure B.99: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

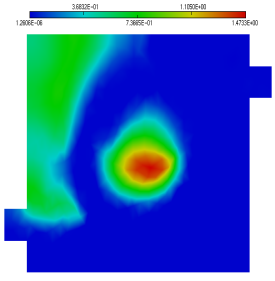

Figure B.100: Computed source with $h = 0.8$, and 1196 triangles.

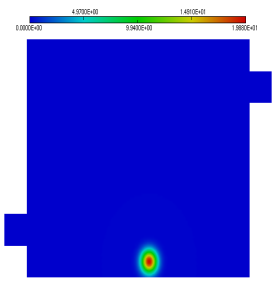

Figure B.101: Actual source.

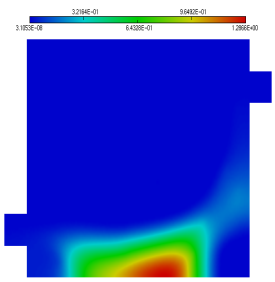

Figure B.102: Computed source with $h = 0.15$ and $31,602$ triangles.

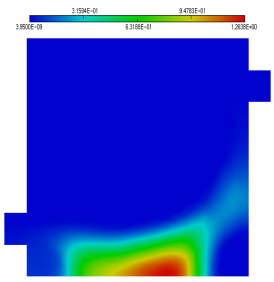

Figure B.103: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

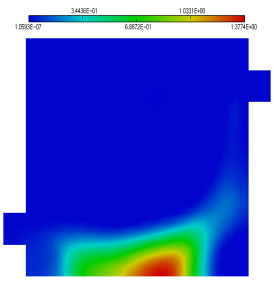

Figure B.104: Computed source with $h = 0.52$, and 2704 triangles.

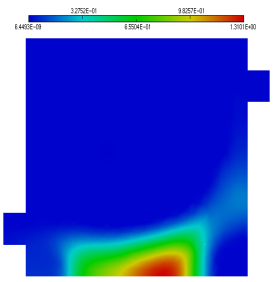

Figure B.105: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

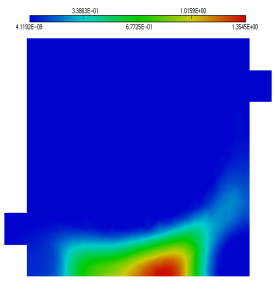

Figure B.106: Computed source with $h = 0.64$, and 1826 triangles.

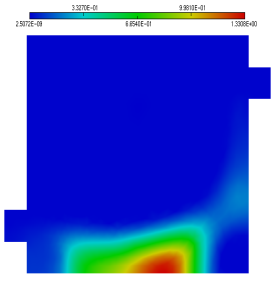

Figure B.107: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

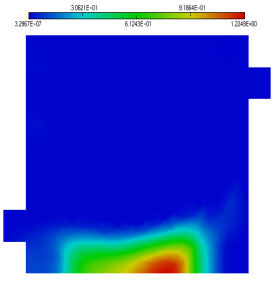

Figure B.108: Computed source with $h = 0.72$, and 1488 triangles.

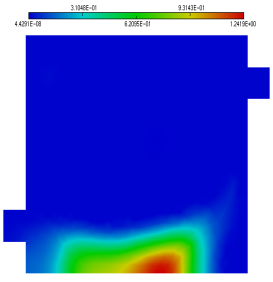

Figure B.109: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

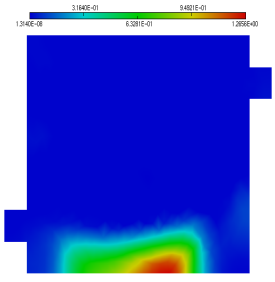

Figure B.110: Computed source with $h = 0.8$, and 1196 triangles.

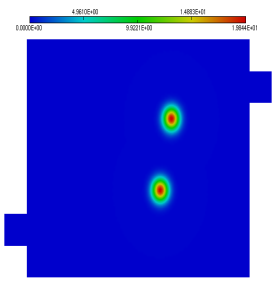

Figure B.111: Actual source.

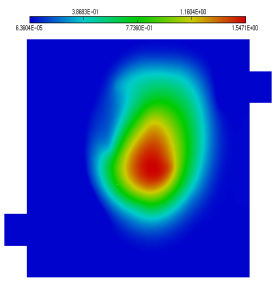

Figure B.112: Computed source with $h = 0.15$ and $31,602$ triangles.

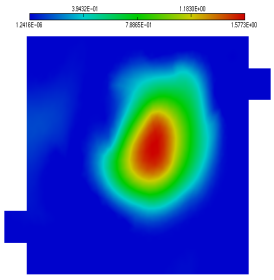

Figure B.113: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

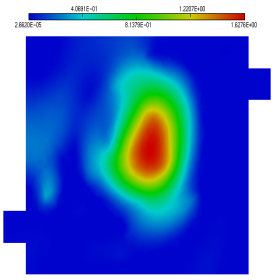

Figure B.114: Computed source with $h = 0.52$, and 2704 triangles.

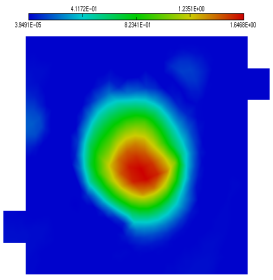

Figure B.115: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

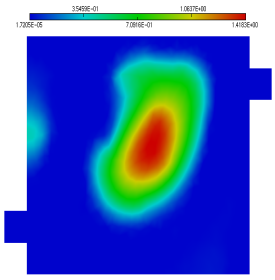

Figure B.116: Computed source with $h = 0.64$, and 1826 triangles.

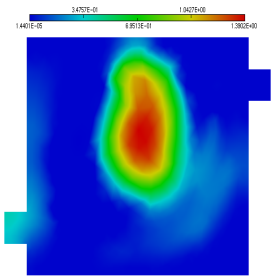

Figure B.117: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

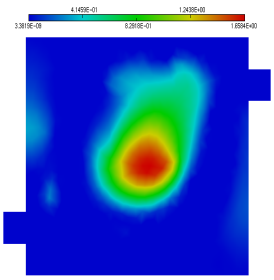

Figure B.118: Computed source with $h = 0.72$, and 1488 triangles.

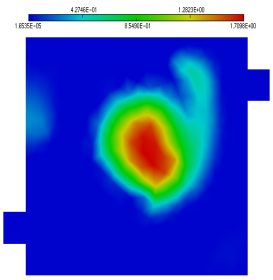

Figure B.119: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

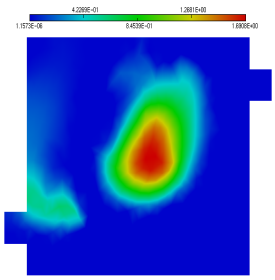

Figure B.120: Computed source with $h = 0.8$, and 1196 triangles.

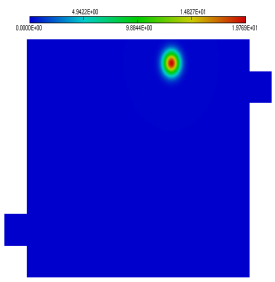

Figure B.121: Actual source.

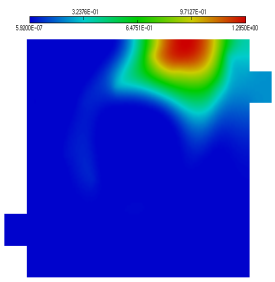

Figure B.122: Computed source with $h = 0.15$ and $31,602$ triangles.

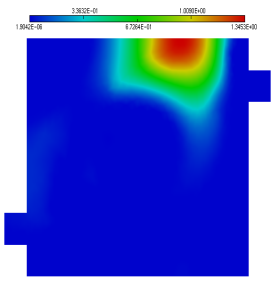

Figure B.123: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

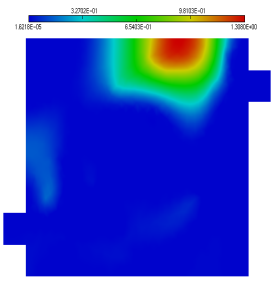

Figure B.124: Computed source with $h = 0.52$, and 2704 triangles.

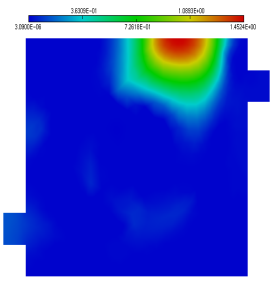

Figure B.125: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

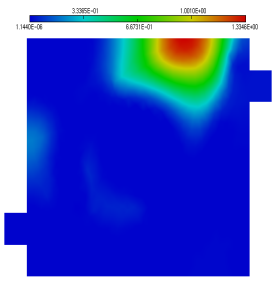

Figure B.126: Computed source with $h = 0.64$, and 1826 triangles.

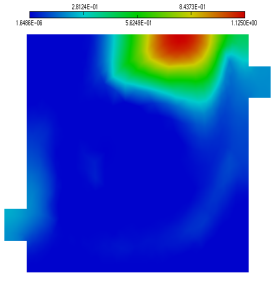

Figure B.127: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

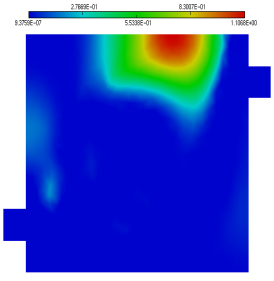

Figure B.128: Computed source with $h = 0.72$, and 1488 triangles.

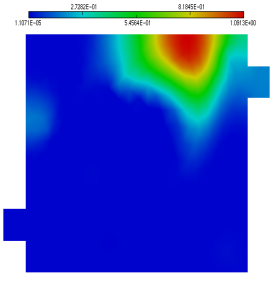

Figure B.129: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

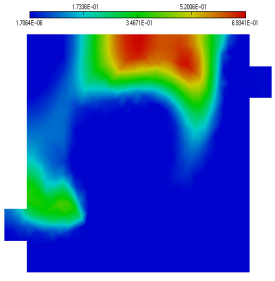

Figure B.130: Computed source with $h = 0.8$, and 1196 triangles.

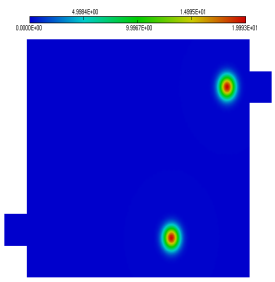

Figure B.131: Actual source.

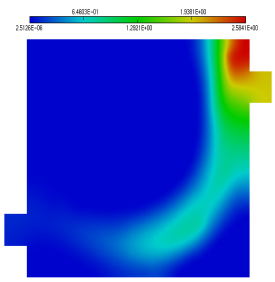

Figure B.132: Computed source with $h = 0.15$ and $31,602$ triangles.

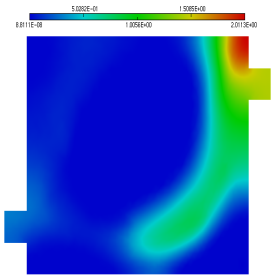

Figure B.133: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

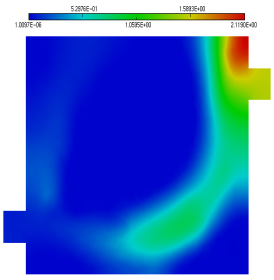

Figure B.134: Computed source with $h = 0.52$, and 2704 triangles.

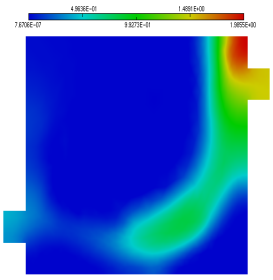

Figure B.135: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

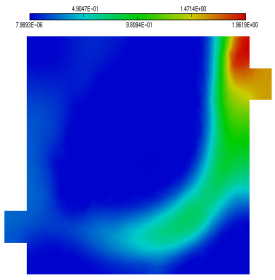

Figure B.136: Computed source with $h = 0.64$, and 1826 triangles.

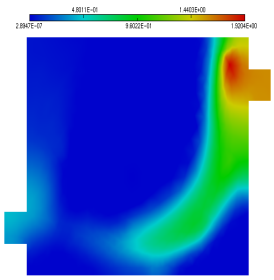

Figure B.137: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

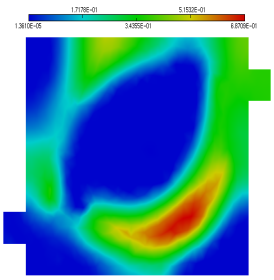

Figure B.138: Computed source with $h = 0.72$, and 1488 triangles.

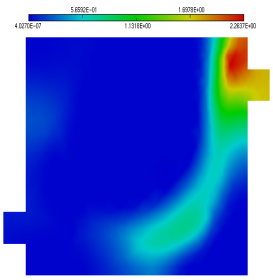

Figure B.139: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

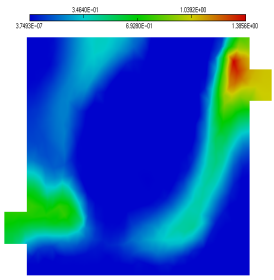

Figure B.140: Computed source with $h = 0.8$, and 1196 triangles.

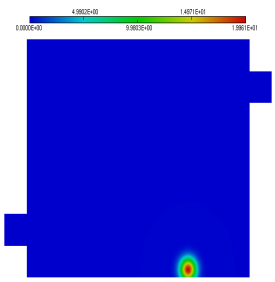

Figure B.141: Actual source.

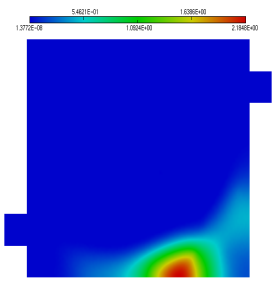

Figure B.142: Computed source with $h = 0.15$ and $31,602$ triangles.

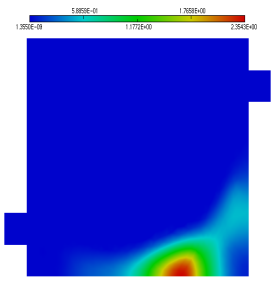

Figure B.143: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

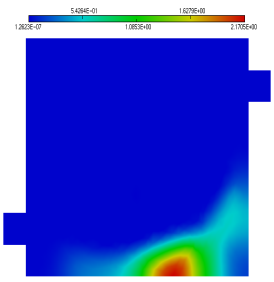

Figure B.144: Computed source with $h = 0.52$, and 2704 triangles.

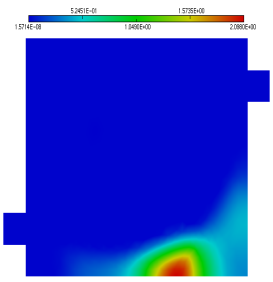

Figure B.145: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

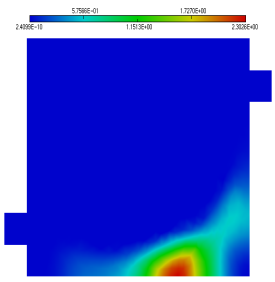

Figure B.146: Computed source with $h = 0.64$, and 1826 triangles.

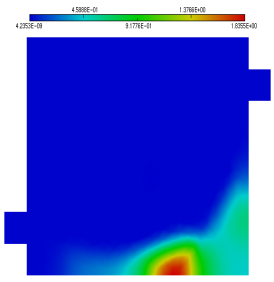

Figure B.147: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

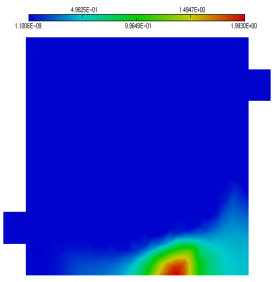

Figure B.148: Computed source with $h = 0.72$, and 1488 triangles.

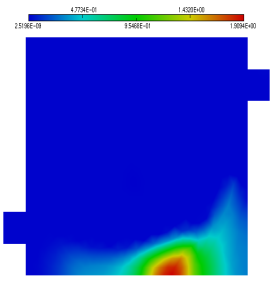

Figure B.149: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

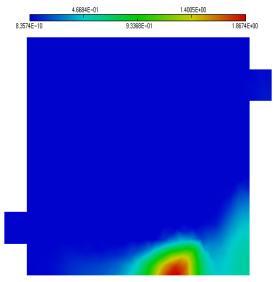

Figure B.150: Computed source with $h = 0.8$, and 1196 triangles.

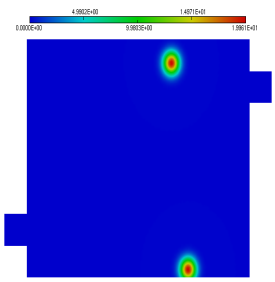

Figure B.151: Actual source.

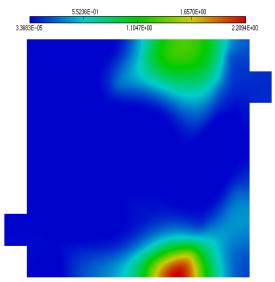

Figure B.152: Computed source with $h = 0.15$ and $31,602$ triangles.

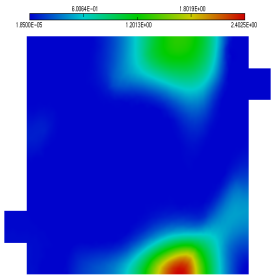

Figure B.153: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

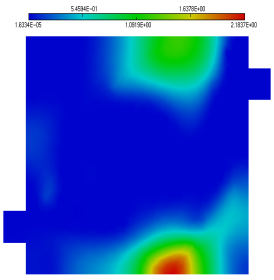

Figure B.154: Computed source with $h = 0.52$, and 2704 triangles.

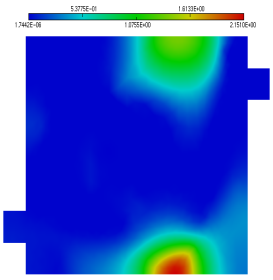

Figure B.155: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

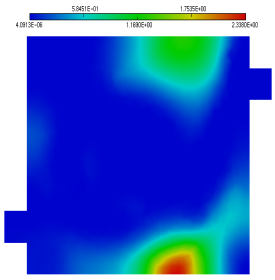

Figure B.156: Computed source with $h = 0.64$, and 1826 triangles.

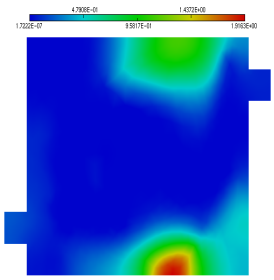

Figure B.157: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

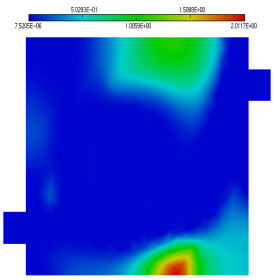

Figure B.158: Computed source with $h = 0.72$, and 1488 triangles.

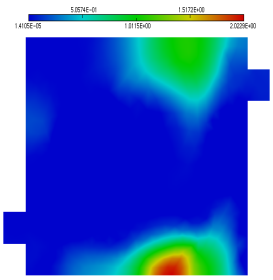

Figure B.159: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

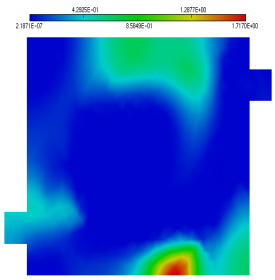

Figure B.160: Computed source with $h = 0.8$, and 1196 triangles.

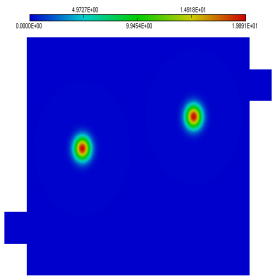

Figure B.161: Actual source.

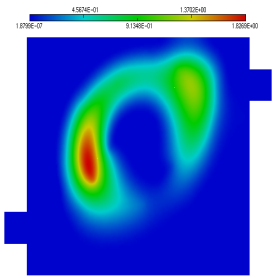

Figure B.162: Computed source with $h = 0.15$ and $31,602$ triangles.

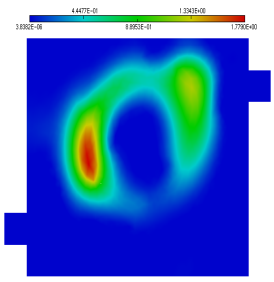

Figure B.163: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

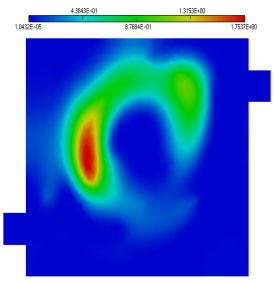

Figure B.164: Computed source with $h = 0.52$, and 2704 triangles.

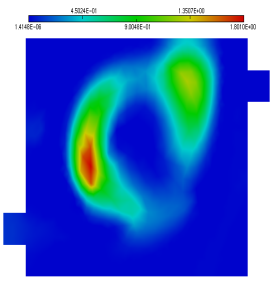

Figure B.165: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

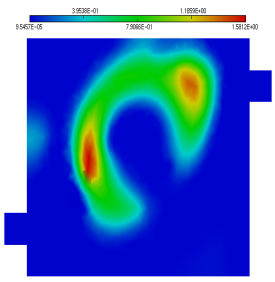

Figure B.166: Computed source with $h = 0.64$, and 1826 triangles.

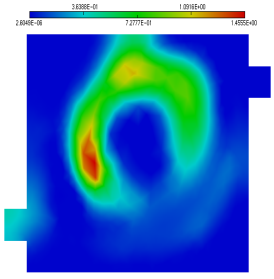

Figure B.167: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

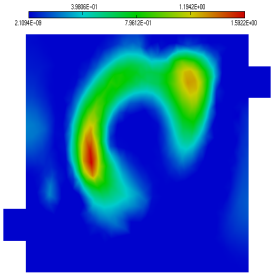

Figure B.168: Computed source with $h = 0.72$, and 1488 triangles.

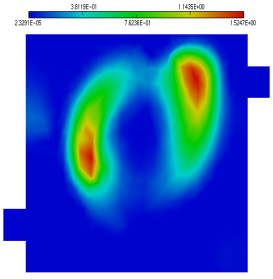

Figure B.169: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

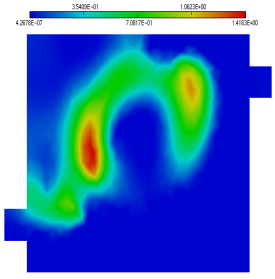

Figure B.170: Computed source with $h = 0.8$, and 1196 triangles.

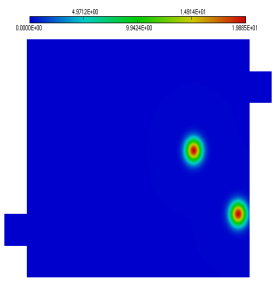

Figure B.171: Actual source.

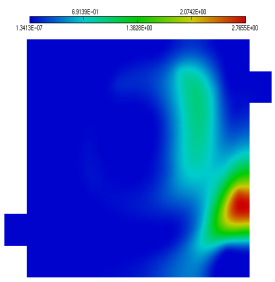

Figure B.172: Computed source with $h = 0.15$ and $31,602$ triangles.

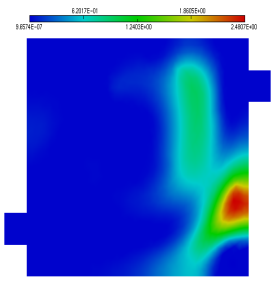

Figure B.173: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

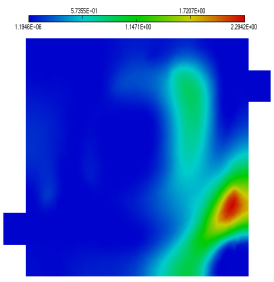

Figure B.174: Computed source with $h = 0.52$, and 2704 triangles.

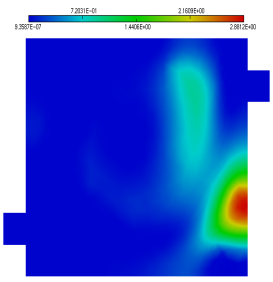

Figure B.175: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

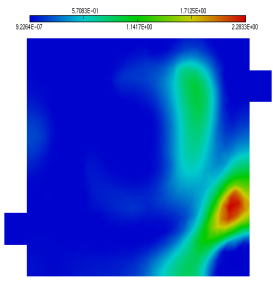

Figure B.176: Computed source with $h = 0.64$, and 1826 triangles.

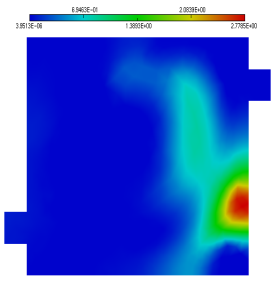

Figure B.177: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

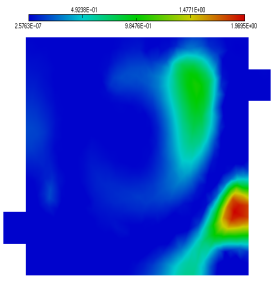

Figure B.178: Computed source with $h = 0.72$, and 1488 triangles.

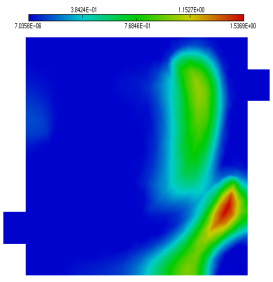

Figure B.179: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

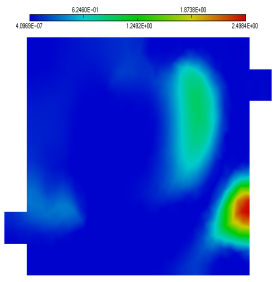

Figure B.180: Computed source with $h = 0.8$, and 1196 triangles.

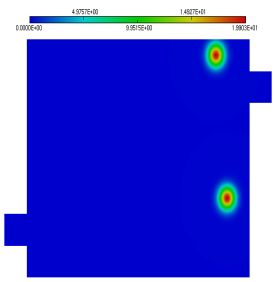

Figure B.181: Actual source.

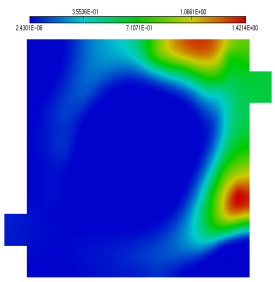

Figure B.182: Computed source with $h = 0.15$ and $31,602$ triangles.

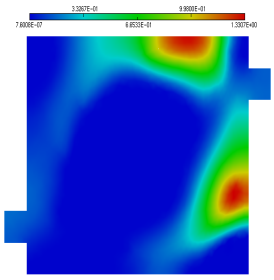

Figure B.183: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

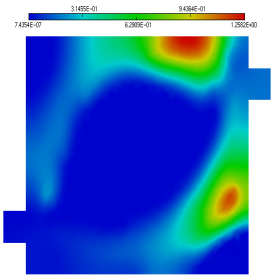

Figure B.184: Computed source with $h = 0.52$, and 2704 triangles.

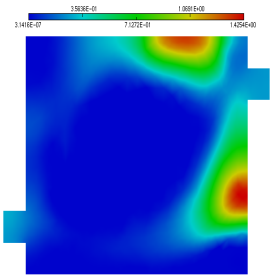

Figure B.185: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

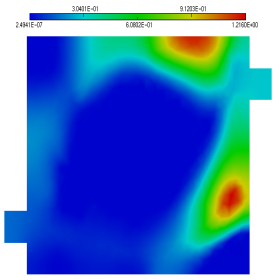

Figure B.186: Computed source with $h = 0.64$, and 1826 triangles.

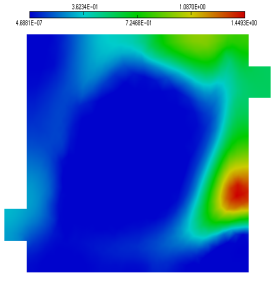

Figure B.187: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

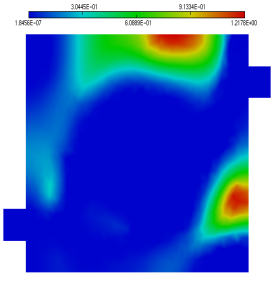

Figure B.188: Computed source with $h = 0.72$, and 1488 triangles.

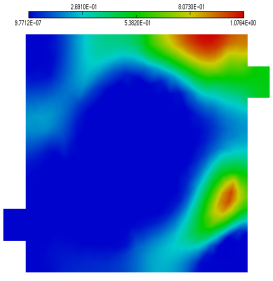

Figure B.189: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

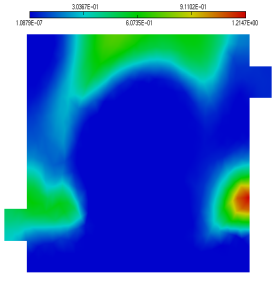

Figure B.190: Computed source with $h = 0.8$, and 1196 triangles.

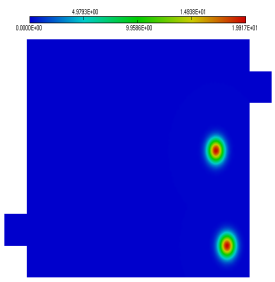

Figure B.191: Actual source.

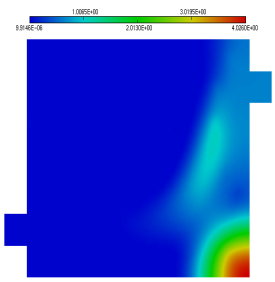

Figure B.192: Computed source with $h = 0.15$ and $31,602$ triangles.

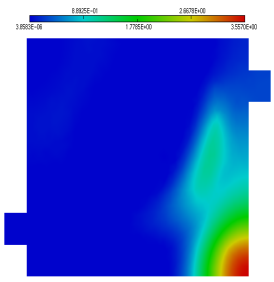

Figure B.193: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

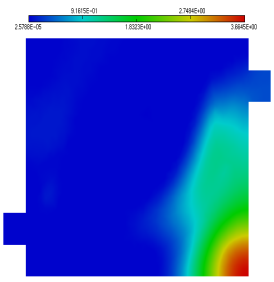

Figure B.194: Computed source with $h = 0.52$, and 2704 triangles.

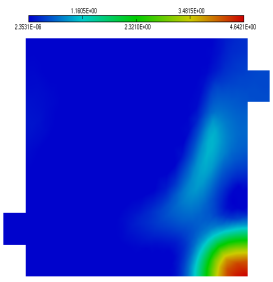

Figure B.195: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

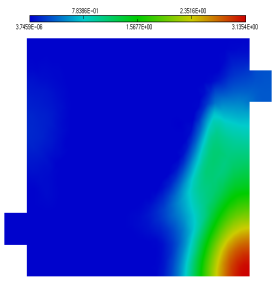

Figure B.196: Computed source with $h = 0.64$, and 1826 triangles.

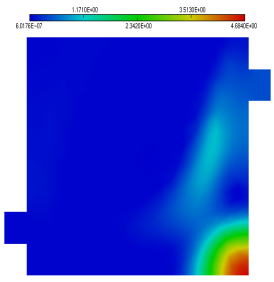

Figure B.197: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

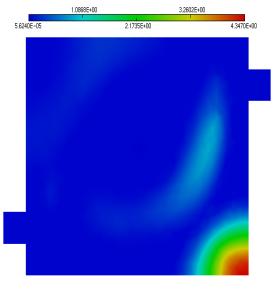

Figure B.198: Computed source with $h = 0.72$, and 1488 triangles.

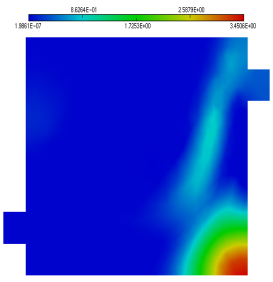

Figure B.199: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

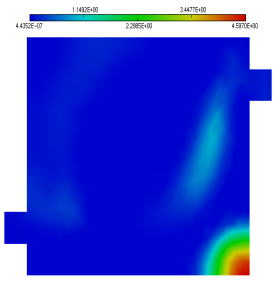

Figure B.200: Computed source with $h = 0.8$, and 1196 triangles.

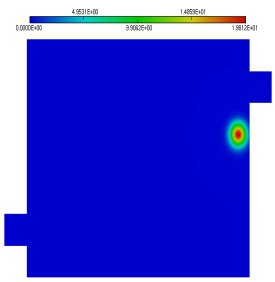

Figure B.201: Actual source.

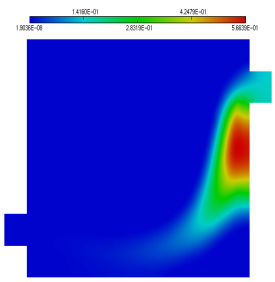

Figure B.202: Computed source with $h = 0.15$ and $31,602$ triangles.

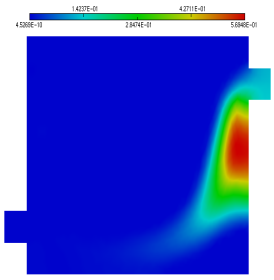

Figure B.203: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

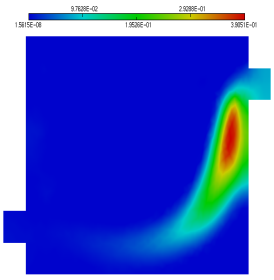

Figure B.204: Computed source with $h = 0.52$, and 2704 triangles.

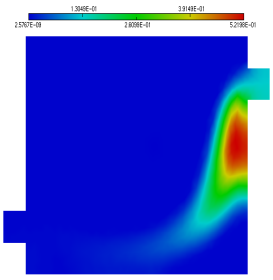

Figure B.205: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

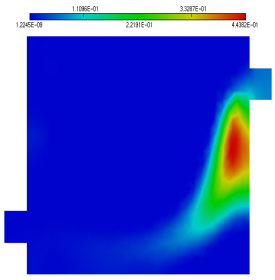

Figure B.206: Computed source with $h = 0.64$, and 1826 triangles.

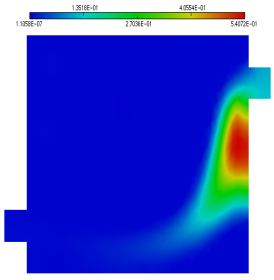

Figure B.207: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

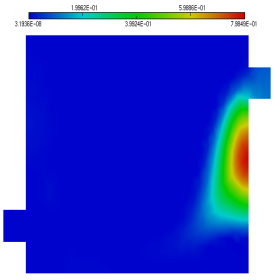

Figure B.208: Computed source with $h = 0.72$, and 1488 triangles.

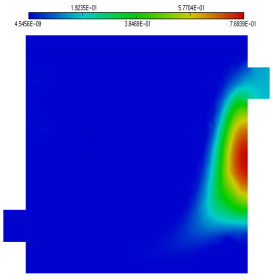

Figure B.209: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

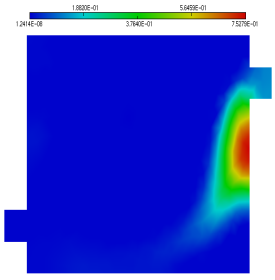

Figure B.210: Computed source with $h = 0.8$, and 1196 triangles.

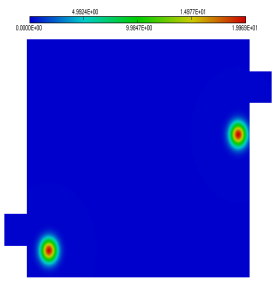

Figure B.211: Actual source.

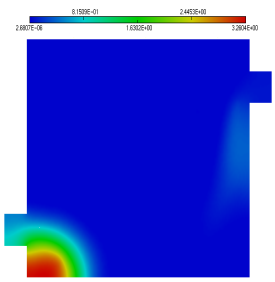

Figure B.212: Computed source with $h = 0.15$ and $31,602$ triangles.

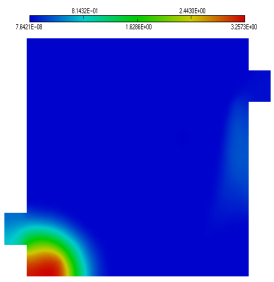

Figure B.213: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

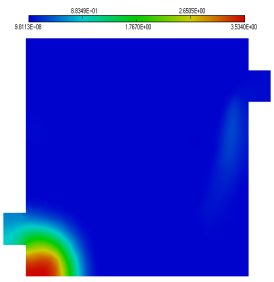

Figure B.214: Computed source with $h = 0.52$, and 2704 triangles.

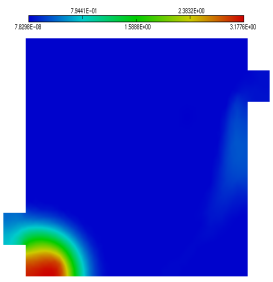

Figure B.215: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

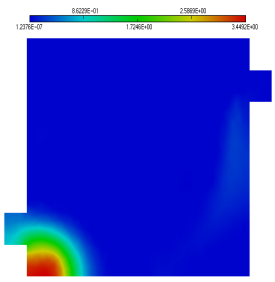

Figure B.216: Computed source with $h = 0.64$, and 1826 triangles.

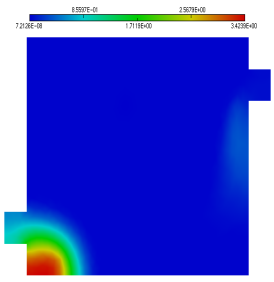

Figure B.217: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

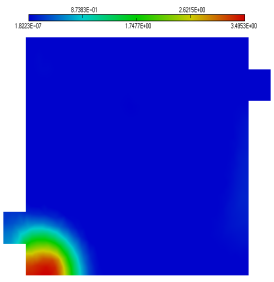

Figure B.218: Computed source with $h = 0.72$, and 1488 triangles.

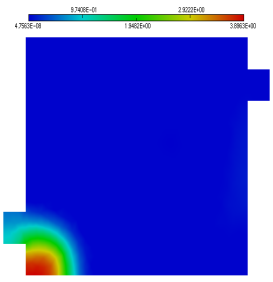

Figure B.219: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

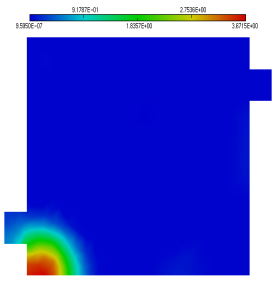

Figure B.220: Computed source with $h = 0.8$, and 1196 triangles.

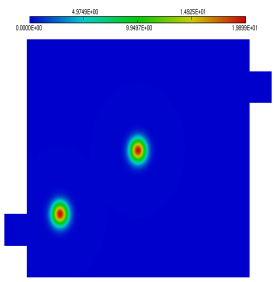

Figure B.221: Actual source.

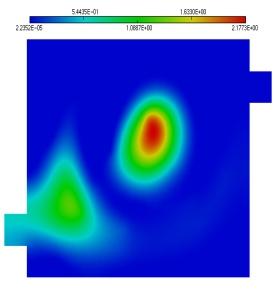

Figure B.222: Computed source with $h = 0.15$ and 31,602 triangles.

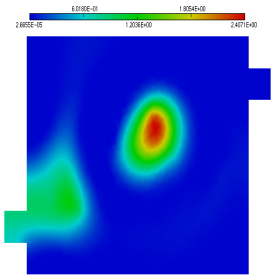

Figure B.223: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

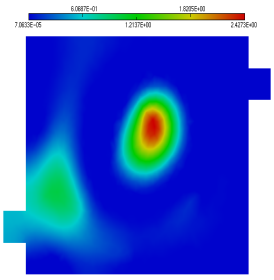

Figure B.224: Computed source with $h = 0.52$, and 2704 triangles.

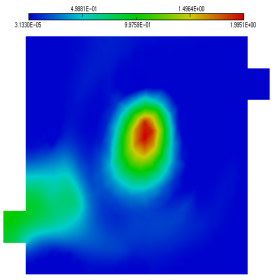

Figure B.225: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

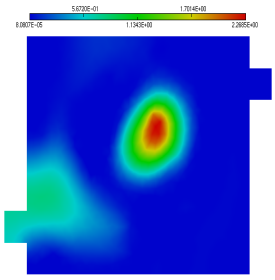

Figure B.226: Computed source with $h = 0.64$, and 1826 triangles.

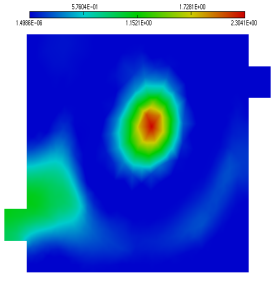

Figure B.227: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

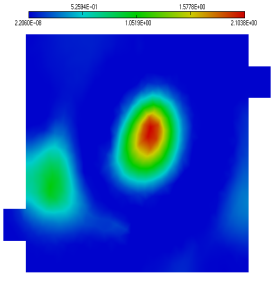

Figure B.228: Computed source with $h = 0.72$, and 1488 triangles.

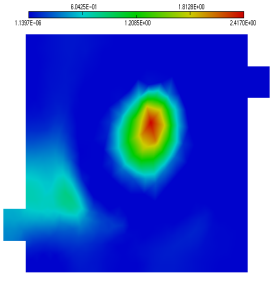

Figure B.229: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

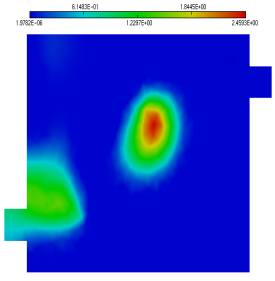

Figure B.230: Computed source with $h = 0.8$, and 1196 triangles.

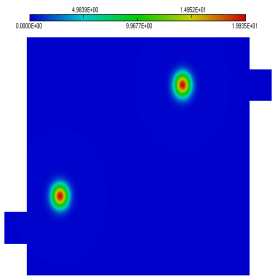

Figure B.231: Actual source.

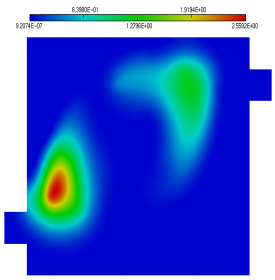

Figure B.232: Computed source with $h = 0.15$ and $31,602$ triangles.

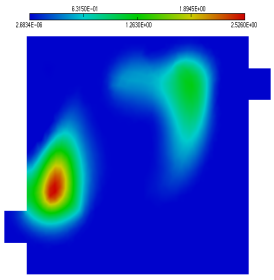

Figure B.233: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

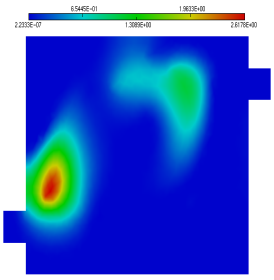

Figure B.234: Computed source with $h = 0.52$, and 2704 triangles.

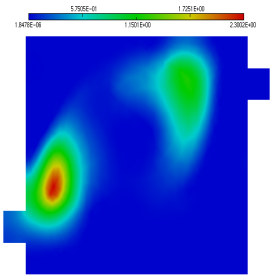

Figure B.235: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

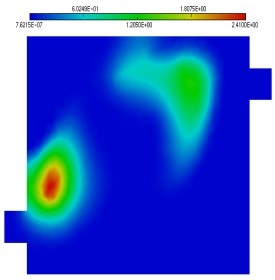

Figure B.236: Computed source with $h = 0.64$, and 1826 triangles.

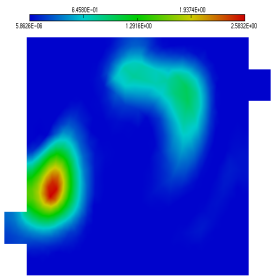

Figure B.237: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

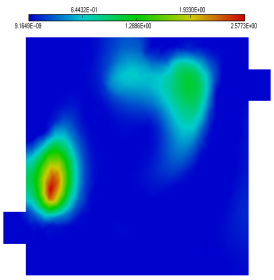

Figure B.238: Computed source with $h = 0.72$, and 1488 triangles.

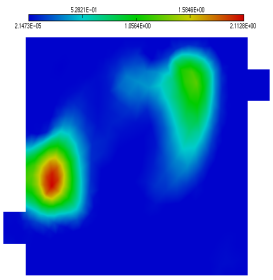

Figure B.239: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

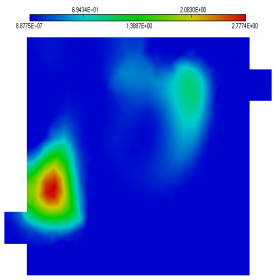

Figure B.240: Computed source with $h = 0.8$, and 1196 triangles.

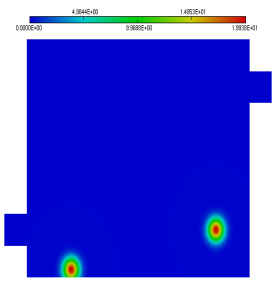

Figure B.241: Actual source.

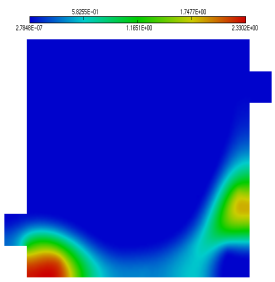

Figure B.242: Computed source with $h = 0.15$ and $31,602$ triangles.

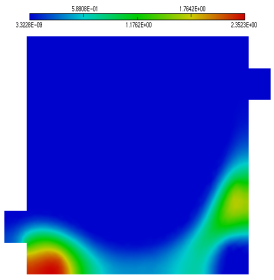

Figure B.243: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

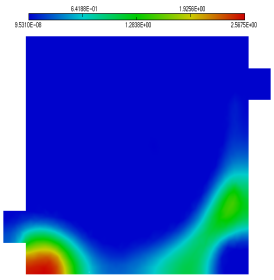

Figure B.244: Computed source with $h = 0.52$, and 2704 triangles.

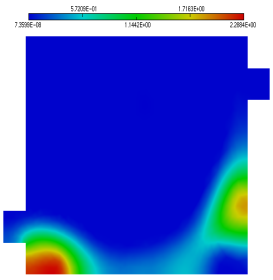

Figure B.245: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

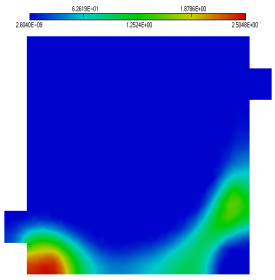

Figure B.246: Computed source with $h = 0.64$, and 1826 triangles.

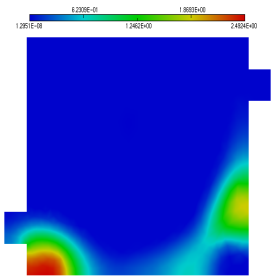

Figure B.247: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

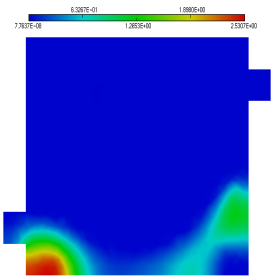

Figure B.248: Computed source with $h = 0.72$, and 1488 triangles.

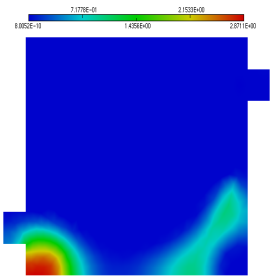

Figure B.249: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

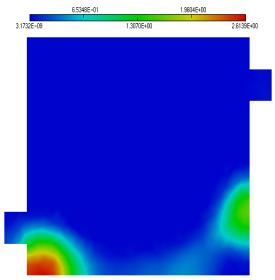

Figure B.250: Computed source with $h = 0.8$, and 1196 triangles.

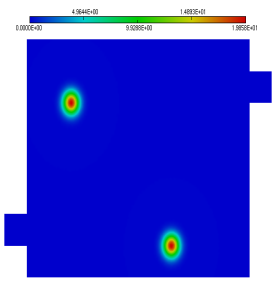

Figure B.251: Actual source.

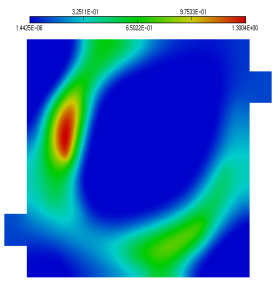

Figure B.252: Computed source with $h = 0.15$ and $31,602$ triangles.

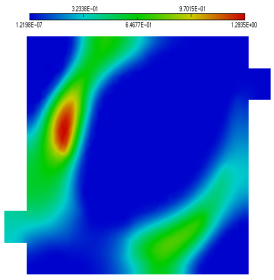

Figure B.253: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

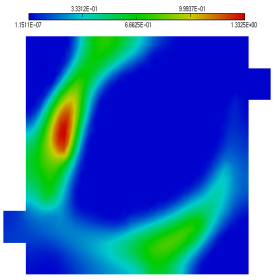

Figure B.254: Computed source with $h = 0.52$, and 2704 triangles.

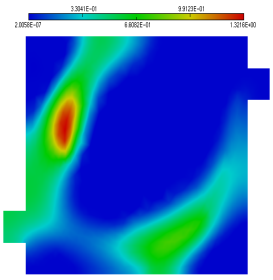

Figure B.255: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

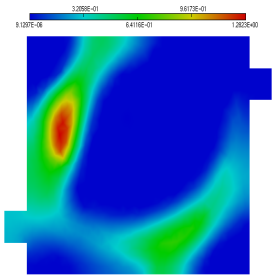

Figure B.256: Computed source with $h = 0.64$, and 1826 triangles.

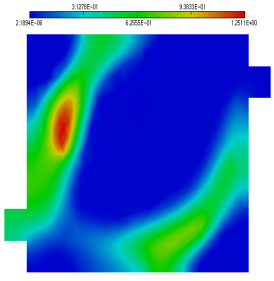

Figure B.257: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

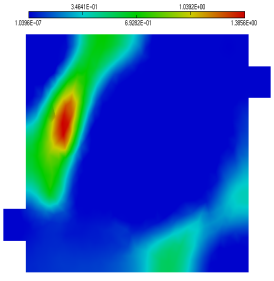

Figure B.258: Computed source with $h = 0.72$, and 1488 triangles.

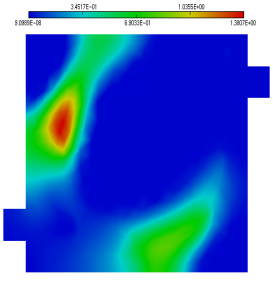

Figure B.259: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

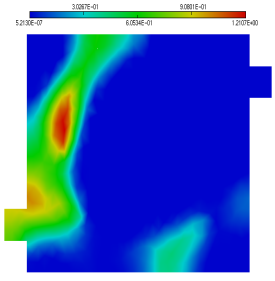

Figure B.260: Computed source with $h = 0.8$, and 1196 triangles.

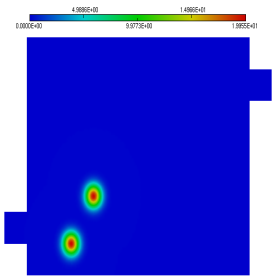

Figure B.261: Actual source.

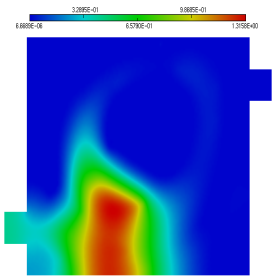

Figure B.262: Computed source with $h = 0.15$ and $31,602$ triangles.

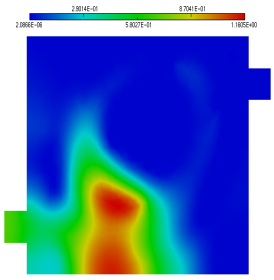

Figure B.263: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

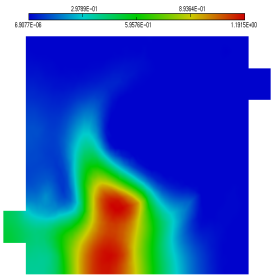

Figure B.264: Computed source with $h = 0.52$, and 2704 triangles.

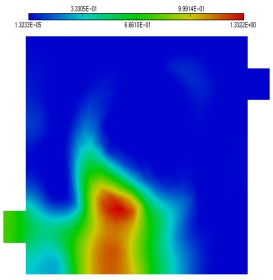

Figure B.265: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

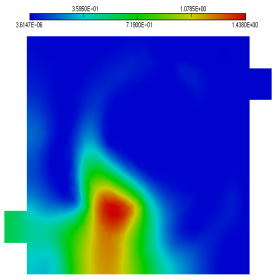

Figure B.266: Computed source with $h = 0.64$, and 1826 triangles.

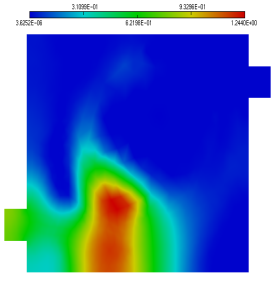

Figure B.267: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

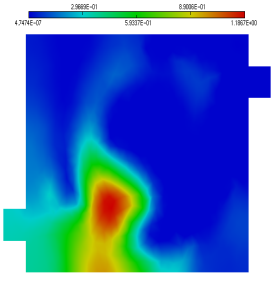

Figure B.268: Computed source with $h = 0.72$, and 1488 triangles.

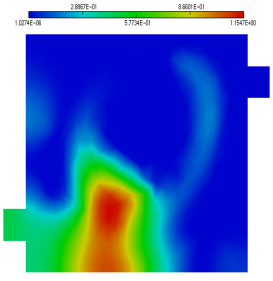

Figure B.269: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

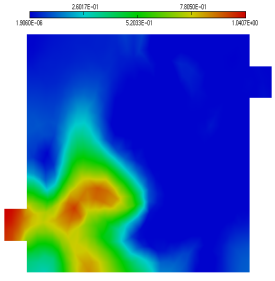

Figure B.270: Computed source with $h = 0.8$, and 1196 triangles.

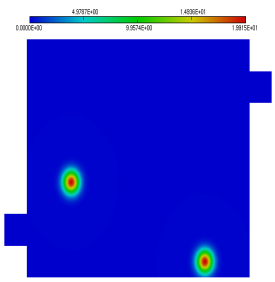

Figure B.271: Actual source.

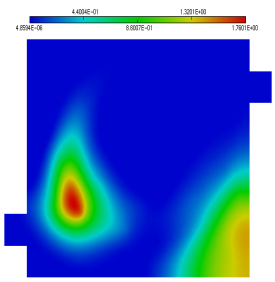

Figure B.272: Computed source with $h = 0.15$ and $31,602$ triangles.

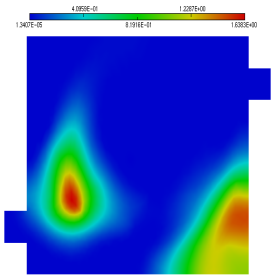

Figure B.273: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

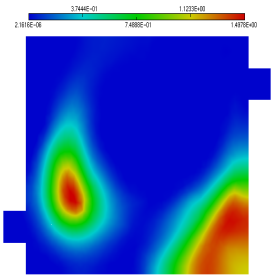

Figure B.274: Computed source with $h = 0.52$, and 2704 triangles.

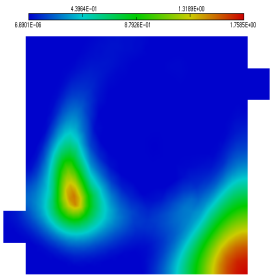

Figure B.275: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

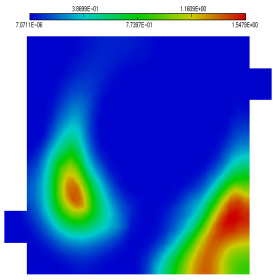

Figure B.276: Computed source with $h = 0.64$, and 1826 triangles.

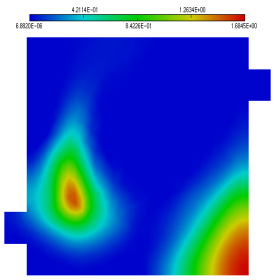

Figure B.277: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

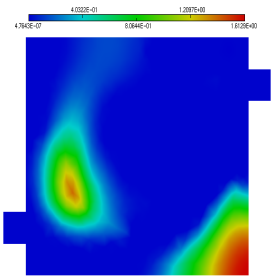

Figure B.278: Computed source with $h = 0.72$, and 1488 triangles.

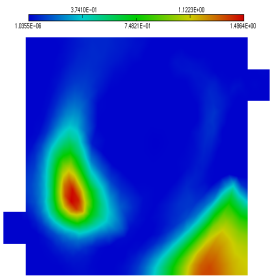

Figure B.279: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

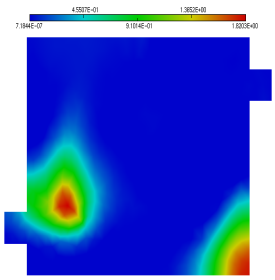

Figure B.280: Computed source with $h = 0.8$, and 1196 triangles.

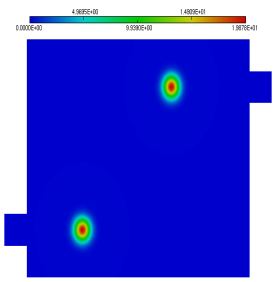

Figure B.281: Actual source.

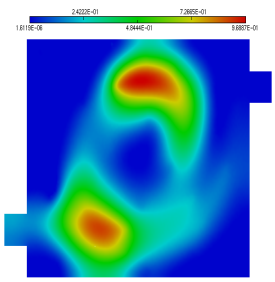

Figure B.282: Computed source with $h = 0.15$ and $31,602$ triangles.

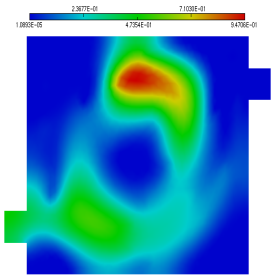

Figure B.283: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

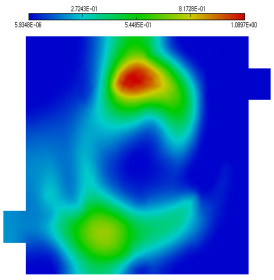

Figure B.284: Computed source with $h = 0.52$, and 2704 triangles.

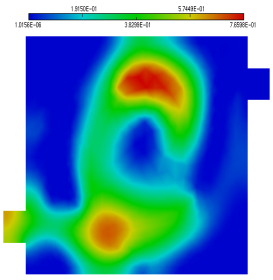

Figure B.285: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

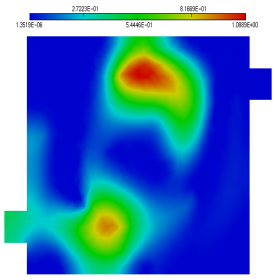

Figure B.286: Computed source with $h = 0.64$, and 1826 triangles.

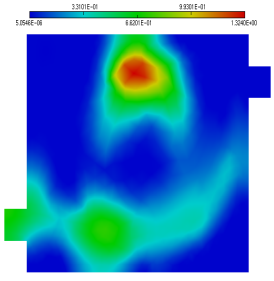

Figure B.287: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

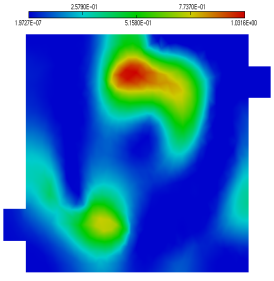

Figure B.288: Computed source with $h = 0.72$, and 1488 triangles.

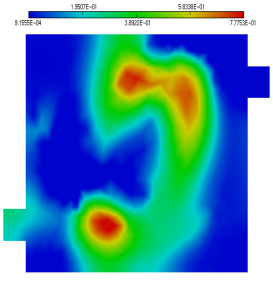

Figure B.289: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

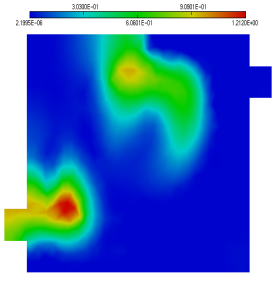

Figure B.290: Computed source with $h = 0.8$, and 1196 triangles.

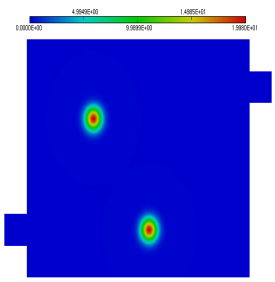

Figure B.291: Actual source.

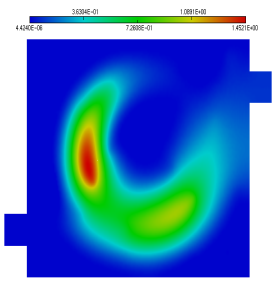

Figure B.292: Computed source with $h = 0.15$ and $31,602$ triangles.

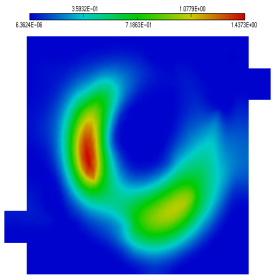

Figure B.293: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

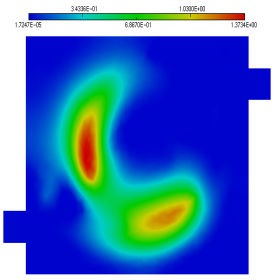

Figure B.294: Computed source with $h = 0.52$, and 2704 triangles.

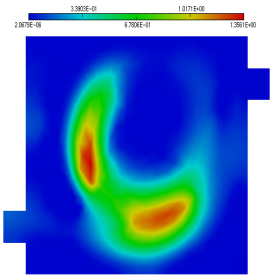

Figure B.295: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

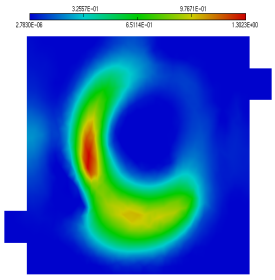

Figure B.296: Computed source with $h = 0.64$, and 1826 triangles.

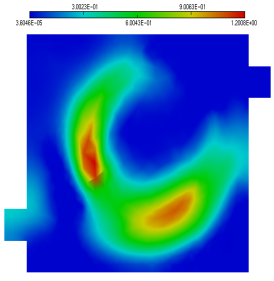

Figure B.297: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

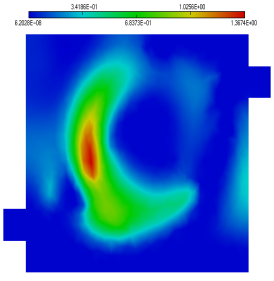

Figure B.298: Computed source with $h = 0.72$, and 1488 triangles.

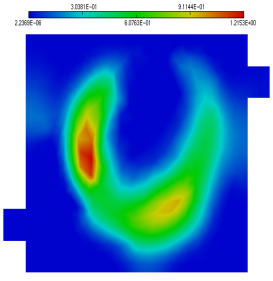

Figure B.299: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

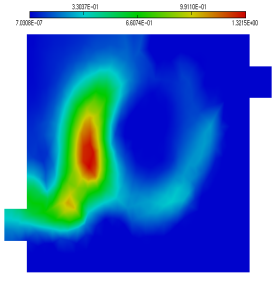

Figure B.300: Computed source with $h = 0.8$, and 1196 triangles.

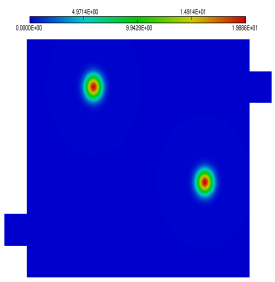

Figure B.301: Actual source.

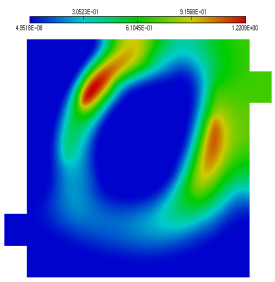

Figure B.302: Computed source with $h = 0.15$ and $31,602$ triangles.

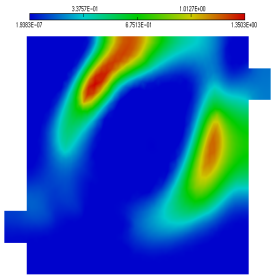

Figure B.303: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

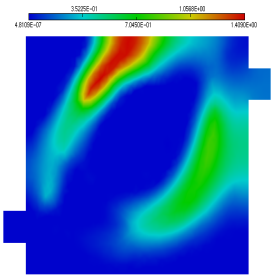

Figure B.304: Computed source with $h = 0.52$, and 2704 triangles.

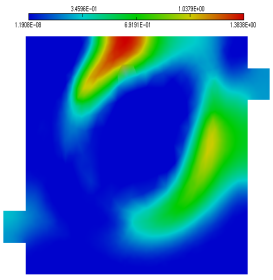

Figure B.305: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

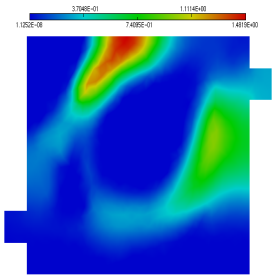

Figure B.306: Computed source with $h = 0.64$, and 1826 triangles.

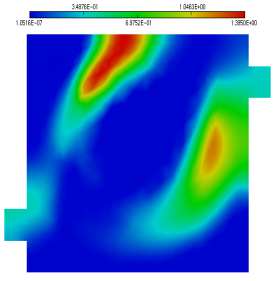

Figure B.307: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

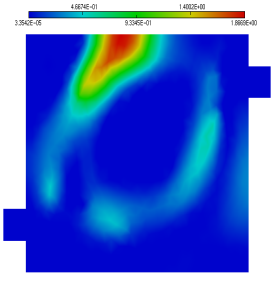

Figure B.308: Computed source with $h = 0.72$, and 1488 triangles.

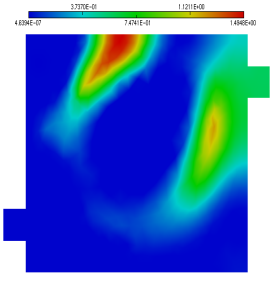

Figure B.309: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

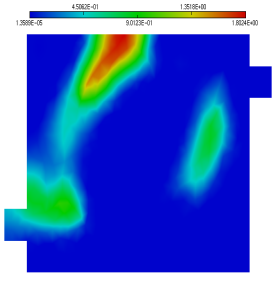

Figure B.310: Computed source with $h = 0.8$, and 1196 triangles.

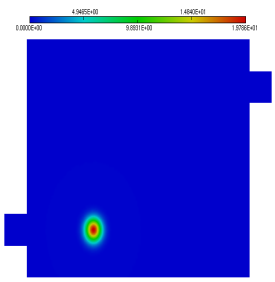

Figure B.311: Actual source.

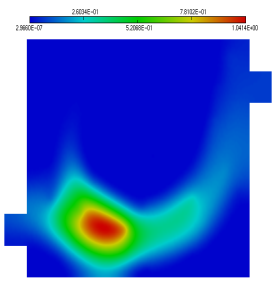

Figure B.312: Computed source with $h = 0.15$ and $31,602$ triangles.

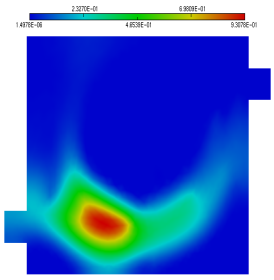

Figure B.313: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

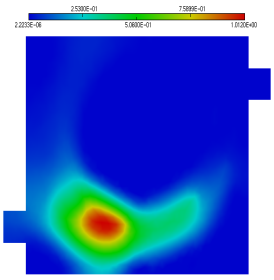

Figure B.314: Computed source with $h = 0.52$, and 2704 triangles.

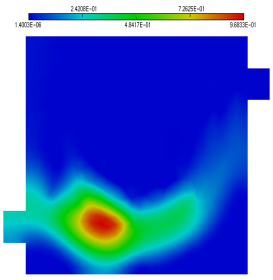

Figure B.315: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

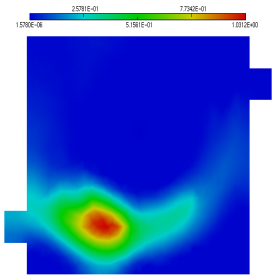

Figure B.316: Computed source with $h = 0.64$, and 1826 triangles.

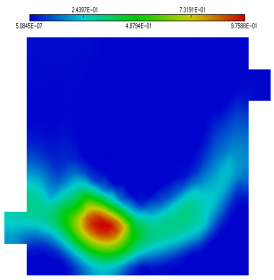

Figure B.317: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

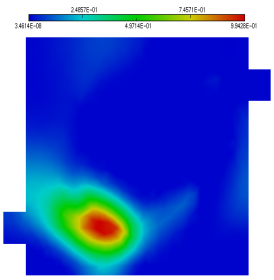

Figure B.318: Computed source with $h = 0.72$, and 1488 triangles.

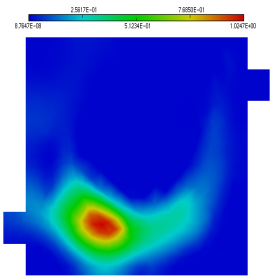

Figure B.319: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

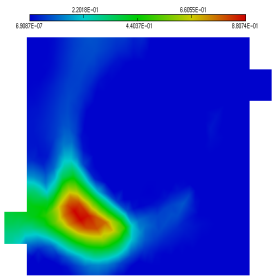

Figure B.320: Computed source with $h = 0.8$, and 1196 triangles.

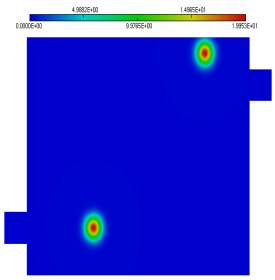

Figure B.321: Actual source.

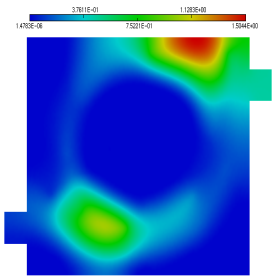

Figure B.322: Computed source with $h = 0.15$ and $31,602$ triangles.

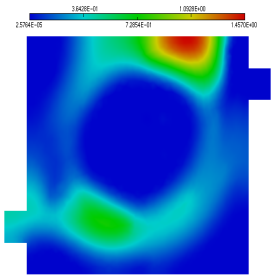

Figure B.323: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

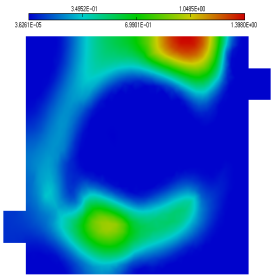

Figure B.324: Computed source with $h = 0.52$, and 2704 triangles.

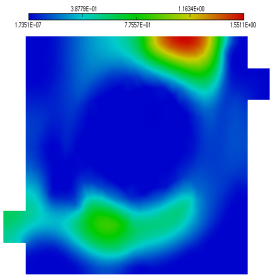

Figure B.325: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

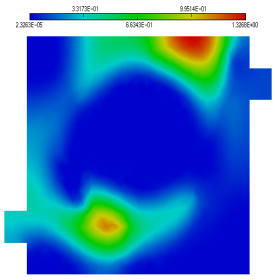

Figure B.326: Computed source with $h = 0.64$, and 1826 triangles.

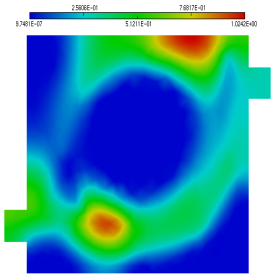

Figure B.327: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

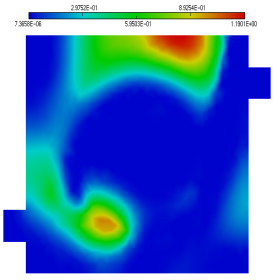

Figure B.328: Computed source with $h = 0.72$, and 1488 triangles.

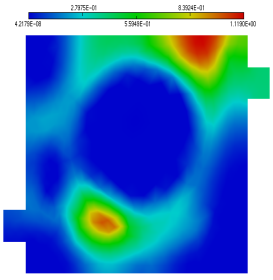

Figure B.329: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

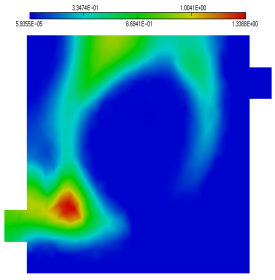

Figure B.330: Computed source with $h = 0.8$, and 1196 triangles.

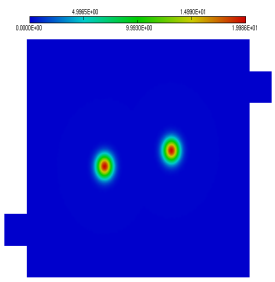

Figure B.331: Actual source.

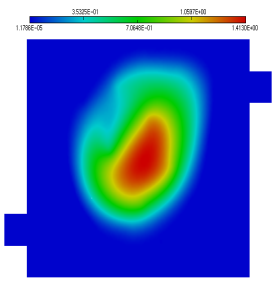

Figure B.332: Computed source with $h = 0.15$ and $31,602$ triangles.

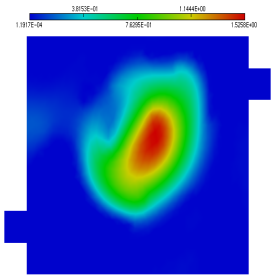

Figure B.333: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

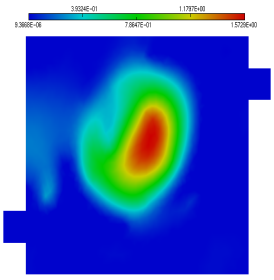

Figure B.334: Computed source with $h = 0.52$, and 2704 triangles.

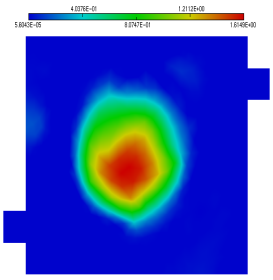

Figure B.335: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

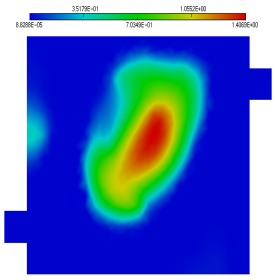

Figure B.336: Computed source with $h = 0.64$, and 1826 triangles.

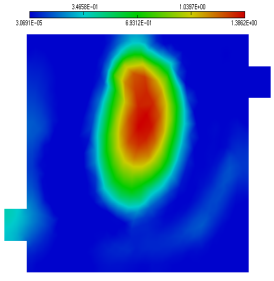

Figure B.337: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

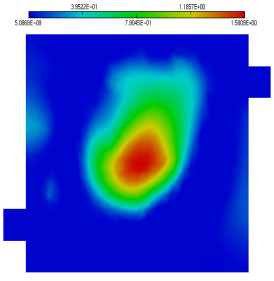

Figure B.338: Computed source with $h = 0.72$, and 1488 triangles.

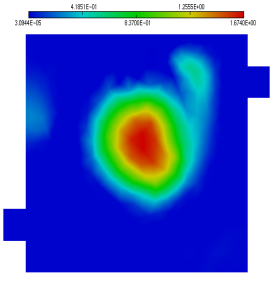

Figure B.339: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

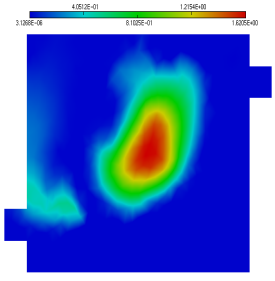

Figure B.340: Computed source with $h = 0.8$, and 1196 triangles.

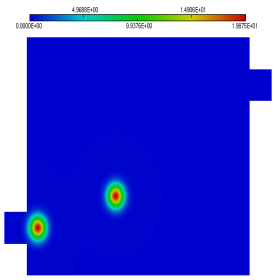

Figure B.341: Actual source.

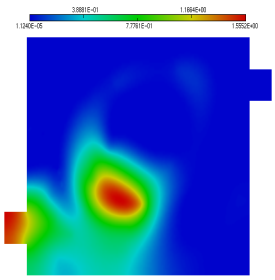

Figure B.342: Computed source with $h = 0.15$ and $31,602$ triangles.

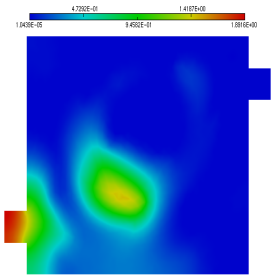

Figure B.343: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

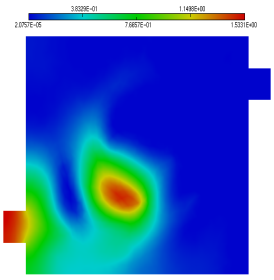

Figure B.344: Computed source with $h = 0.52$, and 2704 triangles.

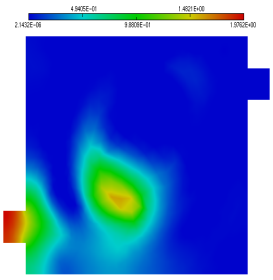

Figure B.345: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

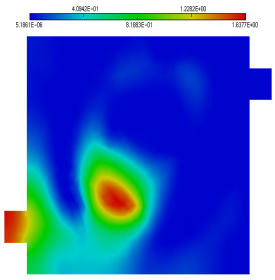

Figure B.346: Computed source with $h = 0.64$, and 1826 triangles.

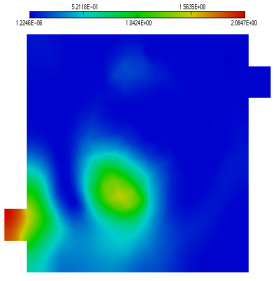

Figure B.347: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

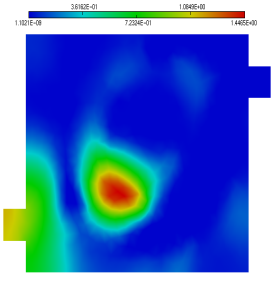

Figure B.348: Computed source with $h = 0.72$, and 1488 triangles.

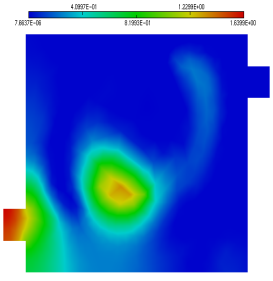

Figure B.349: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

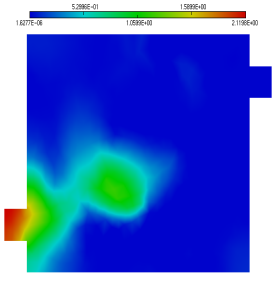

Figure B.350: Computed source with $h = 0.8$, and 1196 triangles.

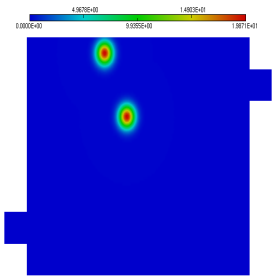

Figure B.351: Actual source.

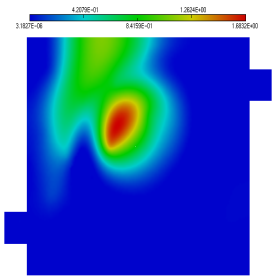

Figure B.352: Computed source with $h = 0.15$ and $31,602$ triangles.

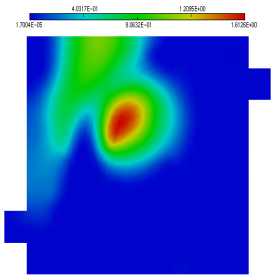

Figure B.353: Computed source with $h_{\max} = 0.6$, $h_{\min} = 0.45$ and 2760 triangles.

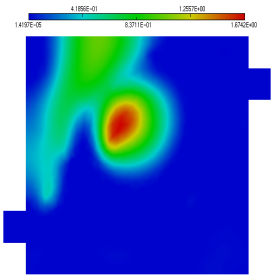

Figure B.354: Computed source with $h = 0.52$, and 2704 triangles.

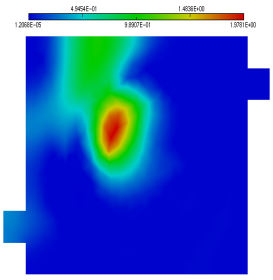

Figure B.355: Computed source with $h_{\max} = 1.0$, $h_{\min} = 0.5$ and 1832 triangles.

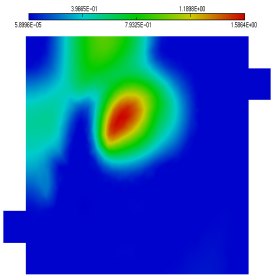

Figure B.356: Computed source with $h = 0.64$, and 1826 triangles.

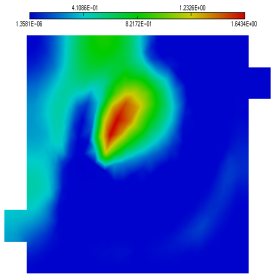

Figure B.357: Computed source with $h_{\max} = 1.1$, $h_{\min} = 0.6$ and 1471 triangles.

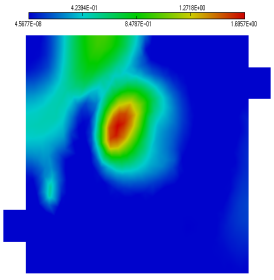

Figure B.358: Computed source with $h = 0.72$, and 1488 triangles.

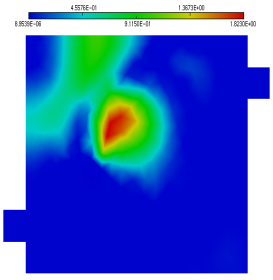

Figure B.359: Computed source with $h_{\max} = 1.2$, $h_{\min} = 0.7$ and 1172 triangles.

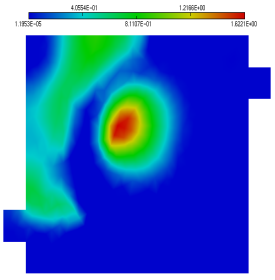

Figure B.360: Computed source with $h = 0.8$, and 1196 triangles.

Bibliography

[1] V. Akcelik, G. Biros, O. Ghattas, K. Long, and B. van Bloemen Waanders. A variational finite element method for source inversion for convective-diffusive transport. 2003 (To appear).

[2] Paul T. Boggs, Paul D. Domich, and Janet E. Rogers. An interior-point method for general large scale quadratic programming problems. *Annals of Operations Research*, 62:419–437, 1996.

[3] Paul T. Boggs, Anthony J. Kearsley, and Jon W. Tolle. A global convergence analysis of an algorithm for large scale nonlinear programming problems. *SIAM Journal on Optimization*, 9(4):833–862, 1999.

[4] Paul T. Boggs, Anthony J. Kearsley, and Jon W. Tolle. A practical algorithm for general large scale nonlinear optimization problems. *SIAM Journal on Optimization*, 9(3):755–778, 1999.

[5] Paul T. Boggs, Kevin R. Long, Stephen B. Margolis, and Patricia A. Howard. Rapid source-inversion for chemical/biological attacks, part 1: The steady-state case. To appear.

[6] Pascal J. Frey. Medit: An interactive mesh visualization software. http://www.inria.fr/rrrt/rt-0253.html, December 2001.

[7] Frédéric Hecht. Bamg: Bidimensional anisotropic mesh generator. http://www-rocq1.inria.fr/gamma/cdrom/www/bamg/eng.htm, October 1998.

[8] K. R. Long. Sundance users manual. Computer science and mathematics research department, Sandia National Laboratories, 2004 (To appear).

[9] Kevin R. Long. Sundance: A rapid prototyping tool for parallel pde-constrained optimization. In L. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, editors, *Large-Scale, PDE-Constrained Optimization*, volume 30 of *Lecture Notes in Computational Science and Engineering*, Heidelberg, 2003. Springer-Verlag.

[10] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.