Inverse Heat Conduction: The Regularized Conjugate Gradient Method with the Adjoint Problem

Victor Minden

April 26, 2012

Contents

1 Motivation **2**

2 Physical Setup **2**
2.1 The Heat Equation 2
2.2 Source and Receivers – The Inverse Problem 4

3 Solution Theory – Nonlinear Conjugate Gradient **5**
3.1 The Gradient 6
3.1.1 The Adjoint Problem 8
3.2 The Conjugation Coefficient 8
3.3 The Step Size 8

4 Numerical Results **9**
4.1 One-Dimensional 9
4.2 Two-Dimensional 10
4.2.1 Regularization 10
4.2.2 Multiple Sensors 11

5 Conclusion **11**

A MATLAB Code **13**

1 Motivation

Inverting the heat equation is a problem of great interest in the sciences and engineering, in particular for modeling and monitoring applications [2]. In this project, we look at the heat equation defined on some domain containing a point heat source at a known location. The magnitude of the heat source is assumed to be unknown and vary with time. Given measurements of the transient temperature at other locations in the domain, the question becomes: can we determine the value of the point source as a function of time?

This is of utility in applications because it allows for determining the temperature of a body at points where direct measurement is infeasible, such as in the case of extremal environments. For example, when a space vehicle reenters the atmosphere, the temperatures experienced by the heat shield can be too large for traditional sensors, and thus we must take measurements at a distance and infer the true temperature [6].

The heat equation we consider is the linear heat equation and, thus, the resulting optimization is that of a linear function. A more general model of heat transfer, however, involves the nonlinear heat equation, which results in a nonlinear optimization problem. We are interested in the solution of such nonlinear problems from an inverse theory standpoint, but the numerical solution of nonlinear partial differential equations is a complicated topic, discussion of which is outside the scope of this project. Thus, we stick to the linear heat equation while exploring algorithms that will work in the general case.

Unfortunately, the inverse problem for this equation is ill-posed, which makes numerical solution difficult without the introduction of regularization or other analysis tools [4]. In particular, we will look at the effects of noise on our reconstruction both with and without a form of Tikhonov-type regularization to promote smoothness of the solution in time.

The conjugate gradient algorithm, regularization technique, and problem formulation described here are not novel work. Rather, they are known methods and a trivial toy probem taken from relevant textbooks [4, 6]. The original contributions presented here are the extension of the problem to multiple sensors[1], the addition of regularization to promote smoothness of the resulting source function estimate[2], and the extension of the problem to two dimensions.

2 Physical Setup

2.1 The Heat Equation

We begin with a brief derivation of the heat equation. While our ultimate simulations of interest will concern the two-dimensional heat equation, we will work here with the one-dimensional equation, noting that the principles are easily applied to the analogous many-dimensional cases. We follow the derivation of Haberman in [1] using conservation of energy and Fourier's law of heat conduction, with some deviation.

[1] Alluded to in [6], but not outlined, nor demonstrated.

[2] Regularization discussed in generality in [3], but not for specific regularization functions, and not the necessary calculations to modify the conjugate gradient algorithm.

Consider a thin insulated rod of length L and cross-sectional area A, assumed without loss of generality to be oriented along the x-axis. We define $e(x,t)$ to be the heat energy density function, which we assume to be constant along a cross-section. If we consider a subsection of the rod, from $x = a$ to $x = b$ (assuming $a < b$), we then see that the total energy in this subsection is

$$\int_a^b e(x,t)A\,dx. \tag{1}$$

From conservation of energy, we can assert that, since the rod is insulated around its circumference, the only way the total heat content in Equation (1) can change is if there is a source or sink $q \in (a,b)$ or there is a heat flux through $x = a$ or $x = b$. We conclude that

$$\frac{d}{dt}\int_a^b e(x,t)A\,dx = \Phi(a,t)A - \Phi(b,t)A + \int_a^b Q(x,t)A\,dx$$

and, thus,

$$\frac{d}{dt}\int_a^b e(x,t)\,dx = \Phi(a,t) - \Phi(b,t) + \int_a^b Q(x,t)\,dx, \tag{2}$$

where, here, $\Phi(x,t)$ is the heat flux from left to right and $Q(x,t)$ is a function representing the net effect of all heat sources and sinks. By Leibniz's rule, we can switch the order of the derivative and the integral on the left-hand side, and by the fundamental theorem of calculus, we know that

$$\Phi(a,t) - \Phi(b,t) = -\left(\Phi(b,t) - \Phi(a,t)\right) = -\int_a^b \frac{\partial \Phi(x,t)}{\partial x}\,dx.$$

Thus, we rewrite Equation (2) as

$$\int_a^b \left[\frac{\partial e(x,t)}{\partial t} + \frac{\partial \Phi(x,t)}{\partial x} - Q(x,t)\right] dx = 0.$$

Because a and b were arbitrary, we know that the above integral must evaluate to zero for any choice of a and b in the domain, which implies that the integrand is identically zero. Therefore,

$$\frac{\partial e(x,t)}{\partial t} + \frac{\partial \Phi(x,t)}{\partial x} - Q(x,t) = 0. \tag{3}$$

We next employ Fourier's law, which states that the heat flux $\Phi(x,t)$ is proportional to the negative spatial derivative of temperature, i.e.,

$$\Phi(x,t) = -K_o \frac{\partial u(x,t)}{\partial x}, \tag{4}$$

where $u(x,t)$ is the temperature function and K_o is the thermal conductivity of the material comprising the rod, which we assume to be constant. Combining Equations (3) and (4), we obtain

$$\frac{\partial e(x,t)}{\partial t} - K_o \frac{\partial^2 u(x,t)}{\partial x^2} - Q(x,t) = 0.$$

To finish the derivation, we relate the energy density $e(x,t)$ to the temperature $u(x,t)$ in order to obtain a differential equation in one function. We note that the $e(x,t)$ depends on two properties of the material – c, the specific heat (which relates the heat energy per unit mass to the temperature), and ρ, the density. Assuming both of these to be constants, simple dimensional analysis shows

$$e(x,t) = c\rho u(x,t),$$

which leads us to our final expression for the one-dimensional heat equation,

$$c\rho\frac{\partial u(x,t)}{\partial t} = K_o\frac{\partial^2 u(x,t)}{\partial x^2} + Q(x,t),$$

or, assuming c, ρ, and K_o to be one[3],

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2} + Q(x,t). \tag{5}$$

In multiple dimensions, it is simple to intuit that the analogous equation of Equation (5) is

$$\frac{\partial u}{\partial t} = \nabla^2 u + Q, \tag{6}$$

where here we have dropped the explicit dependencies of the functions u and Q. Coupled with some appropriate initial and boundary conditions, Equation (6) is the *forward problem*, as we assume knowledge of the sources in the interest of obtaining the final temperature distribution at some time $t_f > 0$.

2.2 Source and Receivers – The Inverse Problem

In contrast to the forward problem of Equation (6), we will not assume knowledge of heat sources inside the domain – at least, not total knowledge. Rather, we'd like to assume that, initially, the domain of interest has a uniformly flat heat profile, which we will assume without loss of generality to be $u(x,0) = 0$. In lieu of any sources or sinks and assuming insulating Neumann boundary conditions, the solution to Equation (6) is simply $u(\mathbf{r},t) = 0$ for all time.

For our setup, following Özisik and Orlande in [6], we are interested in the case of a single source q_s at a known location in the domain, $\mathbf{r}_s$, which has some time-varying strength, i.e.,

$$q_s(\mathbf{r},t) = f(t)\delta(\mathbf{r} - \mathbf{r}_s),$$

where, here, $\delta(\mathbf{r})$ is the Dirac delta function,

$$\delta(\mathbf{r}) = \begin{cases} \infty & \mathbf{r} = 0, \\ 0 & \text{else.} \end{cases}$$

Knowing that the intial heat profile is flat, we could use Equation (6) to predict the effect of $\mathbf{r}_s$ on the temperature distribution u – if we knew $f(t)$. Instead, however, we will look at

[3]Obtainable with a simple change of variables.

the inverse problem: given measurements of u at discrete points in space and time, recover f.

Formally, we consider a number N_r of receivers located at points in the domain distinct from $\mathbf{r}_s$. Using $\mathbf{r}_{r,i}$ to denote the position of the i-th receiver, we assume we have access to measurements of the temperature u at each point $\mathbf{r}_{r,i}$ for N_t different times betwen 0 and t_f.

The problem of interest, then, is the function estimation problem on some domain Ω with boundary $\delta\Omega$:

Given

$$\begin{aligned}
\frac{\partial u(\mathbf{r},t)}{\partial t} &= \nabla^2 u(\mathbf{r},t) + f(t)\delta(\mathbf{r}-\mathbf{r}_s), \quad (\mathbf{r},t) \in \Omega \times (0,t_f) \\
\frac{\partial u(\mathbf{r},t)}{\partial t} &= 0, \quad (\mathbf{r},t) \in \delta\Omega \times (0,t_f) \\
u(\mathbf{r},0) &= 0, \\
u(\mathbf{r}_{r,i},t_j) &= u_{i,j}, \quad i = 1,...,N_r;\ \ j = 1,...,N_t,
\end{aligned} \tag{7}$$

find $f(t)$. We refer to the above as the *inverse problem*. We note, however, that the assumption that we can measure $u(\mathbf{r}_{r,i},t_j)$ exactly is a stronger assumption than one should be willing to make, given that sensor noise / measurement error is always an issue in real-world problems. As such, we will consider the case where we are not given *exactly* $u(\mathbf{r}_{r,i},t_j)$, but, rather, measurements subject to white noise with standard deviation σ. We will see that the impact of this noise on our solution can be quite dramatic and affects our solution algorithm.

3 Solution Theory – Nonlinear Conjugate Gradient

To solve the inverse problem of Equation (7), we will use the nonlinear conjugate gradient method [6], though, of course, other methods exist – most notably, the Levenberg-Marquardt method [4].

In essence, given an initial guess, the conjugate gradient method minimizes a function $J(\mathbf{x})$ by, at each iteration, choosing a new guess by taking the old guess and tacking on an additional term that pushes the solution closer to the optimal one [7]. In contrast to the method of gradient descent, however, the new direction is not required to be strictly in the direction of the gradient, which means that subsequent directions are not constrained to be orthogonal.

The basic idea behind the algorithm is [6], given initial guess $f_k = f_0$,

- Solve for the temperatures, $u_k(\mathbf{r}_{r,i},t_j)$, that result from f_k at the measurement points using the forward problem.
- Calculate the residual error, i.e., the difference between the resultant temperatures, $u_k(\mathbf{r}_{r,i},t_j)$, and the measured temperatures, $u_{i,j}$.
- Find a search direction by "inverting" the heat equation to determine what heat source function would account for the residual error. Potentially modify this search direction by taking into account other known or assumed information about the heat source function[4].

[4]This is where regularization comes in.

- Adjust your guess f_k by adding on some component in the new search direction to form f_{k+1}.
- Set $k = k + 1$ and repeat.

Mathematically, we define $\mathbf{u}_{meas} = \{u_{i,j}\}$ to be the vector of all *measured temperatures* over all times, and $\mathbf{u_f} \equiv H\mathbf{f} = \{u(\mathbf{r}_i, t_j; \mathbf{f})\}$ to be the vector of *calculated temperatures* at all measurement locations and times using the source $\mathbf{f}$, where H is the operator that corresponds to solving the forward problem and vectorizing appropriately. Note the distinction: the first is strictly physical and results from actual measurements, whereas the second is functional and results from computation. We are looking at a concrete, least-squares minimization problem: we want the source $\hat{\mathbf{f}}$ that satisfies

$$\begin{aligned} \hat{\mathbf{f}} &= \underset{\mathbf{f}}{\operatorname{argmin}}\ J[\mathbf{f}], \\ J[\mathbf{f}] &= \frac{1}{2}||\mathbf{u_f} - \mathbf{u}_{meas}||_2^2. \end{aligned} \tag{8}$$

Above, we have adopted the convention of writing the source as a vector rather than a continuous function of time, as we note that any numerical computations will be limited to the discrete setting. We note that the resolution of f may be increased without difficulty (except perhaps computational cost), and thus this in no way limits the applicability of our theory.

We are also highly interested in *regularization* of the inverse problem, where we can employ information known already about $\mathbf{f}$ in some meaningful way. The regularization in which we are interested is Tikhonov approach to promote smoothness of the solution $\hat{\mathbf{f}}$. In particular, if we define D to be a finite-difference matrix that calculates the discrete first derivative, we can modify the cost function in Equation (8) to a regularized cost function,

$$J_r[\mathbf{f}] = \frac{||\mathbf{u_f} - \mathbf{u}_{meas}||_2^2 + \lambda||D\mathbf{f}||_2^2}{2}. \tag{9}$$

In the above, λ is the regularization parameter that represents the trade-off between the data and the *a priori* information – in general, choosing its value is a difficult task, and it frequently must be hand-tuned.

The nonlinear conjugate gradient method, in a more precise algorithmic form than the above, can be seen in Algorithm 1. The variable γ is known as the *conjugation coefficient*, and represents the fact that we want to move in the improved direction, but not so far that the algorithm becomes unstable or oscillates, and β represents the fact that we want to not be a completely memory-less algorithm like gradient descent, but, rather, keep track of the past directions and continue to use that information.

Beyond γ and β, Algorithm 1 requires the computation of the gradient of J at each iteration. We discuss below the theory and method behind computing these values.

3.1 The Gradient

To compute the gradient of the cost function, $\nabla J[\mathbf{f}]$, we will first look at the case without regularization, Equation (8). Cued by Jarny and Özisik in [3], we note that the gradient is

Algorithm 1 Nonlinear Conjugate Gradient (Source: Miller and Karl, [4], with some modifications adapted from [6])

$k = 0$
$\mathbf{f}^{(k)} = \mathbf{f}_0$
$\mathbf{d}^{(k)} = -\nabla J[\mathbf{f}^{(k)}]$
repeat
  Compute $\gamma^{(k)}$
  $\mathbf{f}^{(k+1)} = \mathbf{d}^{(k)} + \gamma^{(k)}\mathbf{f}^{(k)}$
  Compute $\beta^{(k)}$
  $\mathbf{d}^{(k+1)} = -\nabla J[\mathbf{f}^{(k+1)}] + \beta^{(k)}\mathbf{d}^{(k)}$
  $k = k + 1$
until Convergence
$\hat{\mathbf{f}} = \mathbf{f}^{(k)}$

related to the directional derivative by the inner-product,

$$\nabla_{\Delta\mathbf{f}} J[\mathbf{f}] = \langle \nabla J[\mathbf{f}], \Delta\mathbf{f} \rangle , \tag{10}$$

where $\Delta\mathbf{f}$ is some direction in which $\mathbf{f}$ is to be perturbed. An alternative expression which we will use for the directional derivative is simply its definition,

$$\nabla_{\Delta\mathbf{f}} J[\mathbf{f}] \equiv \lim_{\epsilon \to 0} \frac{J[\mathbf{f} + \epsilon\Delta\mathbf{f}] - J[\mathbf{f}]}{\epsilon}.$$

Simple algebra allows us to use the definition to calculate the directional derivative directly,

$$\begin{aligned} \nabla_{\Delta\mathbf{f}} J[\mathbf{f}] &= \lim_{\epsilon \to 0} \frac{||H(\mathbf{f} + \epsilon\Delta\mathbf{f}) - \mathbf{u}_{meas}||_2^2 - ||H\mathbf{f} - \mathbf{u}_{meas}||_2^2}{2\epsilon} \\ &= \left\langle H^T(H\mathbf{f} - \mathbf{u}_{meas}), \Delta\mathbf{f} \right\rangle . \end{aligned}$$

By equating the above with Equation (10) and noting that the choice of direction was arbitrary, we can assert that

$$\nabla J[\mathbf{f}] = H^T(H\mathbf{f} - \mathbf{u}_{meas}), \tag{11}$$

for the case of no regularization. Because the gradient is additive, it is simple to see that adding regularization as in Equation (9) yields

$$\begin{aligned} \nabla J_r[\mathbf{f}] &= \nabla J[\mathbf{f}] + \lambda\nabla\left(\frac{||D\mathbf{f}||_2^2}{2}\right) \\ &= H^T(H\mathbf{f} - \mathbf{u}_{meas}) + \lambda D^T D\mathbf{f}. \end{aligned}$$

The problem now becomes determining what is meant by the adjoint operators H^T and D^T. As D is the time-derivative operator, it is easy to show[5] that the adjoint operator is the negative time-derivative, i.e., $D^T = -D$.

[5]Use integration by parts in an L_2 functional inner-product, or go straight to the discrete difference matrices and observe the transpose. Note that we must assume appropriate boundary conditions, at $t = 0$ and $t = t_f$.

The application of the adjoint heat equation operator (with measurement), H^T, is more complicated, and a derivation is outside the scope of this project. We below give the statement of the operator H^T without proof.

3.1.1 The Adjoint Problem

In [3], Jarny and Özisik describe the adjoint problem stated here. As H is an operator that takes source-function space to measurement space, H^T should, naturally, be a function that takes measurement space to source-function space. The final form of the adjoint operator involves the solution to the adjoint problem,

$$\begin{aligned} -\frac{\partial \psi(\mathbf{r},t)}{\partial t} &= \nabla^2 \psi(\mathbf{r},t) + \sum_{i=1}^{N_r} \tilde{u}_i(t)\delta(\mathbf{r}-\mathbf{r}_{r,i}), \quad (\mathbf{r},t) \in \Omega \times (0,t_f) \\ \frac{\partial \psi(\mathbf{r},t)}{\partial t} &= 0, \quad (\mathbf{r},t) \in \delta\Omega \times (0,t_f) \\ \psi(\mathbf{r},t_f) &= 0, \end{aligned} \tag{12}$$

where $\tilde{u}_i(t)$ is the continuous function achieved by time-interpolation of the measurements from sensor i. We note that Equation (12) is a *final*-value problem, but can be converted to an initial-value problem by making time go backwards [3].

The definition of H^T, then, is simply,

$$H^T \mathbf{u} = \psi,$$

where ψ is given by solving the adjoint problem with the temperature information in $\mathbf{u}$ broken apart once again into separate sensors and then interpolated in time.

3.2 The Conjugation Coefficient

There are several heuristics for calculating the conjugation coefficient, γ, including, but not limited to, the Fletcher-Reeves method, and the Polak-Ribière method [4]. We use the latter, the expression for which is

$$\gamma^{(k)} = \frac{-\left\langle \nabla J[\mathbf{f}^{(k+1)}], \nabla J[\mathbf{f}^{(k+1)}] - \nabla J[\mathbf{f}^{(k)}] \right\rangle}{||\nabla J[\mathbf{f}^{(k)}]||_2^2}. \tag{13}$$

3.3 The Step Size

Combined, γ and ∇J are all we need to determine the next search direction. The next variable to be determined is the optimal step size in that direction, β.

To determine the optimal β for the stepsize, we need to calculate how the output is perturbed as a response to a perturbation in the input of the forward problem of Equation (6). Thus, we are interested in the following *sensitivity problem* [3],

$$\begin{aligned} \frac{\partial \Delta u(\mathbf{r},t)}{\partial t} &= \nabla^2 \Delta u(\mathbf{r},t) + \Delta f(t)\delta(\mathbf{r}-\mathbf{r}_s), \quad (\mathbf{r},t) \in \Omega \times (0,t_f) \\ \frac{\partial \Delta u(\mathbf{r},t)}{\partial t} &= 0, \quad (\mathbf{r},t) \in \delta\Omega \times (0,t_f) \\ \Delta u(\mathbf{r},t_f) &= 0. \end{aligned} \tag{14}$$

Using this to solve for Δu given Δf, we can that compute $\beta^{(k)}$ by differentiating $J[\mathbf{f}^{(k+1)}]$ with respect to β and equating that to zero. This yields [6]

$$\beta^{(k)} = \frac{\left\langle \mathbf{u}_{\mathbf{f}}^{(k)} - \mathbf{u}_{meas}, \Delta \mathbf{u}^{(k)} \right\rangle}{||\Delta \mathbf{u}^{(k)}||_2^2}, \tag{15}$$

where, once again, $\mathbf{u}_{\mathbf{f}}^{(k)}$ is the vectorized measurements from all sensors across time.

4 Numerical Results

For all of the results below, we use a simple finite-difference discretization to solve the forward problem. In particular, we use an implicit discretization in time and a simple central difference in space, enforcing Neumann conditions at the spatial boundaries by assuming ghost points outside the domain. The focus of this project is not the method of solution of the involved PDEs, and, thus, we have not taken great length to ensure the most efficient method of solution, and have not included time-to-solution results.

For each simulation, the following parameters were used:

- Maximum allowable iterations: 200
- Minimum required reduction in residual per iteration: 0.1%
- Tolerable residual size: 1E-12

As a parameter, maximum iterations is self-explanatory. For the minimum required residual reduction, we simply monitored the percent decrease in the cost function at each iteration – if that percent decrease ever got below 0.1%, we terminate the iteration. Finally, the residual size was checked at each iteration to determine if the cost function had been brought down to a sufficiently low value. In practice, we always exited based off of the minimum required residual reduction as opposed to the other exit parameters. This corresponds to simply exiting the algorithm when it stops giving any real improvement, and taking whatever output we get at that point as the solution.

4.1 One-Dimensional

We first look at the case of the one-dimensional heat equation with a single sensor and no noise. We consider the domain $x \in [0, 1]$ and $t \in (0, t_f = 1]$, with the heat source located at $x = 0.5$ and sensor located at $x = 0.9$. We take measurements at the sensor every millisecond., giving us 1000 measurements with which to work.

Figure 1 show the function estimation results of the heat source in 1D with no noise. We see that the estimate is very good for most of the time range (in fact, for times $t < 0.7$, the error is bounded by 2E-2), but deteriorates near the end of the temporal domain. It is noted in [6] that the reconstruction algorithm we are using enforces the right boundary to have a value of zero, and, in some ways, this makes sense. Our measurements include data up to the

time $t = t_f$, but we realize that the heat equation requires some non-zero time to propagate[6]. Thus, given that the sensor is physically separated from the source, a measurement at time t_f does not reflect any information about the source's behavior at time t_f. Thus, we assume that the source has attenuated to zero at that time in the solution algorithm, which upsets estimation in cases where that behavior is violated.

[Figure 1 about here.]

4.2 Two-Dimensional

In two-dimensions, we will look at both regularization and varying the number of sensors, both with and without measurement noise. We consider the unit square, and, once again, $t \in (0, t_f = 1]$. In all cases, the source is located at $(0.5, 0.5)$.

[Figure 2 about here.]

We first place a single sensor at the point $(0.8, 0.8)$, again taking measurements every millisecond. In Figure 2, we see that the output looks largely the same as the 1D case, which isn't completely surprising – we have a point heat source on a symmetric domain.

[Figure 3 about here.]

When we add noise, things start to fall apart for this simple physical setup. In Figure 3 we see the effects of adding measurement noise with a signal-to-noise ratio of 1000 (i.e., the mean measured signal is 1000 times greater than the noise variance). As can be seen in the figure, the estimate is completely off and noise dominates.

4.2.1 Regularization

To recover the original signal from above, we first attempt the Tikhonov regularization previously described. Hand-tuning the algorithm led us to a regularization parameter of $\alpha =$ 1E-3. In Figure 4, we see the result of this regularization. While the signal is still noisy, it is much closer to the actual signal of interest than the non-regularized solution of Figure 3 is. We can make out the actual features of the signal, which is a great improvement, and the error stays within reasonable levels (barring the right boundary, which we previously discussed).

[Figure 4 about here.]

Trials with alternative regularization parameters showed great variety in the solution qualiy, showing the sensitivity of this regularization to the parameter α. In particular, we noted that making α larger caused the solution to be very smooth and mostly ignore the data, as can be seen in Figure 5.

[Figure 5 about here.]

[6]Though we remark that for any time $t > t_0$, the heat information from time t_f will have propogated throughout the entire region, in the mathematically ideal model [1].

4.2.2 Multiple Sensors

A natural extension from the one-sensor case is to use multiple sensors. Assuming the measurement noise is uncorrelated across sensors, having more data should lead us to a better solution. We modify the regularized problem from above to use four sensors arranged in a square around the source, with vertices at $\{0.2, 0.8\}^2$.

As can be seen in Figure 6, the additional sensors do, in fact, give us more information about the source, which in turn leads to a cleaner reconstruction. We note that, without regularization, using multiple sensors did not provide a solution that was substantially better than the noise-flooded one of Figure 3. This is probably due to the fact that, even with many measurements, a large amount of noise just makes it too difficult to determine what the actual object is unless you use some prior information (such as smoothness).

[Figure 6 about here.]

As a final test, we increased the sampling time of the sensors from every millisecond to every 0.1 milliseconds, while continuing to sample at the same four points with the same signal-to-noise ratio. We see in Figure 7 that the additional information obtained is very helpful, and the high-frequency error is damped to the point that the quality of the reconstruction is almost par with that of the no noise case of Figure 2. There is a definite low-frequency mode in the error of this last reconstruction, but our regularization method isn't designed to damp low-frequency modes, so there's not much that can be done in this framework.

[Figure 7 about here.]

5 Conclusion

Our results showed that the nonlinear conjugate gradient method proves to be an effective algorithm for function estimation given the gradient of the cost function – provided the correct cost function is chosen. Without regularization, the reconstructions were very sensitive to noise, and did not resemble the original signal.

With regularization on the smoothness of the source function, however, the solutions were much more recognizable and exhibited less error. Adding multiple sensors to the mix caused the error to decrease further, yielding solutions which very closely resembled the true signal, with some artifacts (potentially due to the regularization). After regularization, the primary source of error was due to the non-zero propogation time of the heat profile, causing the solution to be error-prone near the end of the simulation time-frame (where less and less information could be gained from the measurements). If one is interested in the later times, one simple solution suggested in [6] is to take measurements for a longer duration, thus pushing the error out of the region of interest. In applications where the temperature is to be measured continuously and we wish to update the source estimation in real-time (for example, using a sliding data window), this would simply correspond to a small delay in the estimation system, a "linear-phase response", so to speak.

To more closely examine this problem, it would be nice to perform simulations on the full non-linear heat equation, as well as accounting for heterogeneous domains. In addition,

more intelligent methods of solving the underlying PDE would allow us to perform realistic run-time analysis to determine the comparative efficiency of this algorithm.

One potential method of improving this algorithm as it stands would be to use a known heuristic to calculate the regularization parameter α. While there is no one heuristic that always works, there are several the have strong statistical justification which can yield good results [4].

An interesting extension of this algorithm would be to drop some of the assumptions. For example, core to this application thus far has been the assumption that the heat source is a point source, and that we know its location. If we assumed to not know the location of the heat source, we then have an extra degree of freedom which could greatly complicate the result. If the function is not assumed to be a point source but perhaps to have some known physical distribution (maybe we are interested in the temperature distribution of some component of a machine with known size, shape, and location), the algorithm becomes similarly complicated.

In sum, this project investigated the heat equation in multiple dimensions and explored a method for solving the inverse source problem with and without regularization. Regularization was found to be instrumental in obtaining accurate results, demonstrating the value of such methods in practice. The final results obtained exhibited low noise and the algorithm proved robust in multiple trials.

References

[1] R. Haberman, *Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems*, Pearson Prentice Hall, 2004.

[2] Y. Hon and T. Wei, *A fundamental solution method for inverse heat conduction problem*, Engineering Analysis with Boundary Elements, 28 (2004), pp. 489 – 495.

[3] Y. Jarny, M. Ozisik, and J. Bardon, *A general optimization method using adjoint equation for solving multidimensional inverse heat conduction*, International Journal of Heat and Mass Transfer, 34 (1991), pp. 2911 – 2919.

[4] E. Miller and W. Karl, *Fundamentals of Inverse Problems*, Not yet published, 2012.

[5] W. B. Muniz, H. F. de Campos Velho, and F. M. Ramos, *A comparison of some inverse methods for estimating the initial condition of the heat equation*, Journal of Computational and Applied Mathematics, 103 (1999), pp. 145 – 163.

[6] M. Ozisik and H. Orlande, *Inverse Heat Transfer: Fundamentals and Applications*, Taylor & Francis, 2000.

[7] L. Trefethen and D. Bau, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997.

A MATLAB Code

```
%We assume initial condition is all 0, and BCs are insulated (Neumann, 0
%derivative).
clear all;
close all;
clc;

%X domain is [0,1], step size is hx
hx = 1e-1;
x  = 0:hx:1;

%t domain is (0,tMax], step size is ht
tMax = 1;
ht   = 1e-4;
t    = 0:ht:tMax;
t    = t(2:end);

%Measurements are taken at mLoc, point source is at sLoc
mLoc = [0.2 0.2; 0.8 0.8; 0.2 0.8; 0.8 0.2];
%mLoc = [0.8 0.8];
sLoc = [0.5, 0.5];

%Need to give a source function and take measurements
%(construct true solution)
TRUE_SOURCE =sin(5*t).^2;
TRUE_SOURCE = TRUE_SOURCE';
TRUE_TMEAS  = solveDirect(x,t,mLoc,sLoc,TRUE_SOURCE);

%Add noise
sigma2 = mean(mean(TRUE_TMEAS))*0.001;
TRUE_TMEAS_BACKUP = TRUE_TMEAS;
TRUE_TMEAS = TRUE_TMEAS + sqrt(sigma2).*randn(size(TRUE_TMEAS));

%%
%Regularization parameter
alpha = 1e-3;
%alpha = 0.25;
%First we need an initial guess for the source, which is located at sLoc.
s     = zeros(length(t),1);
gradSOld = ...
    solveAdjoint(x,t,mLoc,sLoc,zeros(length(t),size(mLoc,1)),TRUE_TMEAS);
d = 0;
tol = 1e-12;
maxits = 200;
oldcost = 1e8;
Δtol = 1e-3;
%%
for i = 1:maxits
%Solve direct problem --> get T
Tmeas = solveDirect(x,t,mLoc,sLoc,s);

```

```
%Check cost
cost  = costFunction(Tmeas,TRUE_TMEAS);
if (cost < tol || abs(cost - oldcost)/oldcost < Δtol)
    break
end

%Solve adjoint problem given T at the measurement location(s)
%and the measured data --> get lambda

lambda = solveAdjoint(x,t,mLoc,sLoc,Tmeas,TRUE_TMEAS);

%With lambda, compute grad(S)
gradS = lambda - alpha*[diff(s,2);0;0];

%With grad(S), compute gamma (how much weight on old direction) and new
%direction
gamma = computeGamma(gradS,gradSOld,i);
d     = gradS + gamma*d;

%Solve sensitivity problem in this new direction
ΔT = solveSensitivity(x,t,mLoc,sLoc,d);

%compute step size beta
betaCoeff = computeBeta(Tmeas(:),TRUE_TMEAS(:),ΔT(:));

%compute new source guess, repeat
s = s - betaCoeff*d;
gradSOld = gradS;
oldcost = cost;
end
%%
```

```
function Tmeas = solveDirect(x,t,mLoc,sLoc,s)
%Solve the direct problem with insulated boundaries.
%Input: x (space vector), t (time vector), measurement location, source
%location, source function
%Output: temperature at each timestep evaluated at the measurement
%location.

hx    = x(2) - x(1);
ht    = t(2) - t(1);
Nx    = length(x);

TOld  = zeros(Nx*Nx,1);

%Construct second derivative matrix
D      = ht/hx^2*(diag(-2*ones(Nx,1)) + diag(ones(Nx-1,1),1) + ...
    diag(ones(Nx-1,1),-1));

%Fix BCs
D(1,1)       = -ht/hx^2;
D(1,2)       = ht/hx^2;
```

```
20
21 D(end,end)   = -ht/hx^2;
22 D(end,end-1) = ht/hx^2;
23 D = kron(eye(Nx),D) + kron(D,eye(Nx));
24
25
26 D = eye(size(D)) - D;
27 D = sparse(D);
28 %Vectors to hold result
29 Tmeas  = zeros(length(t),size(mLoc,1));
30 sourceIndex = zeros(size(sLoc,1));
31 measureIndex = zeros(size(mLoc,1));
32 for i = 1:size(sLoc,1)
33     sourceIndexX = find(abs(x-sLoc(i,1))<hx/10);
34     sourceIndexY = find(abs(x -sLoc(i,2))<hx/10);
35     sourceIndex(i) = MNtoI(sourceIndexX,sourceIndexY,Nx,Nx);
36 end
37 for i = 1:size(mLoc,1)
38     measureIndexX = find(abs(x-mLoc(i,1)) < hx/10);
39     measureIndexY = find(abs(x-mLoc(i,2))<hx/10);
40     measureIndex(i) = MNtoI(measureIndexX,measureIndexY,Nx,Nx);
41 end
42
43 for k = 1:length(t)
44    TNew       = D\TOld;
45
46    %Add source
47    for i = 1:size(sLoc,1)
48     TNew(sourceIndex(i)) = TNew(sourceIndex(i)) + ht*s(k,i);
49
50    %Grab T at mLoc
51    for i = 1:size(mLoc,1)
52     Tmeas(k,i)    = TNew(measureIndex(i));
53    end
54
55    %Update TOld
56    TOld = TNew;
57 end
58
59 end
```

```
1 function lambda = solveAdjoint(x,t,mLoc,sLoc,estimateT,trueT)
2 %This needs to be a change-of-variables on the direct equation
3     force = 2*(estimateT-trueT);
4     force = flipud(force);
5     lambda = solveDirect(x,t,sLoc,mLoc,force);
6     lambda = flipud(lambda);
7 end
```

```
1 function ΔT = solveSensitivity(x,t,mLoc,sLoc,source)
2 ΔT = solveDirect(x,t,mLoc,sLoc,source);
```

```
end
```

```
function [cost] = costFunction(estimateT,trueT)
    cost = (norm(estimateT-trueT))^2;
end
```

```
function beta = computeBeta (Tmeas,TRUE_TMEAS,ΔT)
beta = (Tmeas - TRUE_TMEAS)'*ΔT / (ΔT'*ΔT);
end
```

```
function gamma = computeGamma(gradS,gradSOld,k)
gamma = 0;
if k == 1
    return
end
gamma = gradS'*(gradS - gradSOld) / (gradSOld'*gradSOld);
end
```

List of Figures

1 The 1D problem with no noise. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom). 18

2 The 2D problem with no noise and one sensor. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom). 18

3 The 2D problem with noise and one sensor. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom). 19

4 The 2D problem with noise, one sensor, and regularization with α = 1E-3. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom). 20

5 The 2D problem with noise, one sensor, and regularization with α = 1E-2. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom). 20

6 The 2D problem with noise, four sensors, and regularization. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom). 21

7 The 2D problem with noise, four sensors, regularization, and temperature samples every 0.1 milliseconds. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom). 21

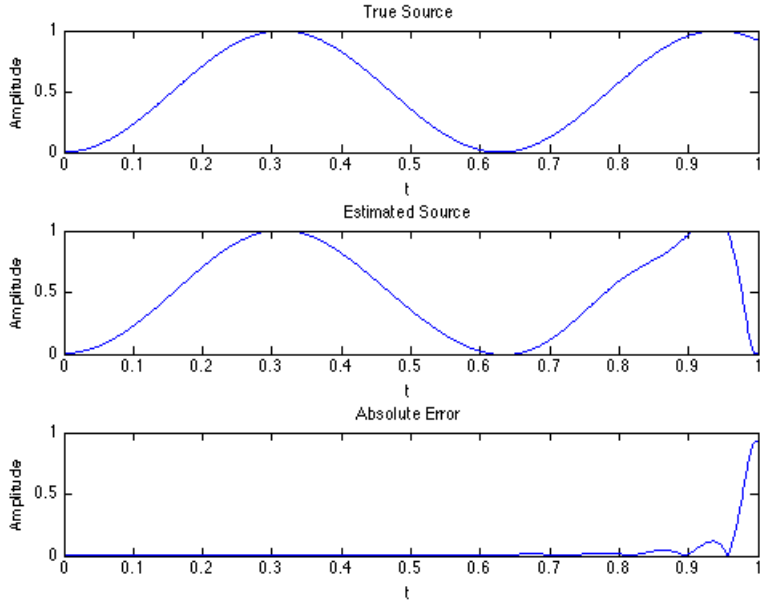

Figure 1: The 1D problem with no noise. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom).

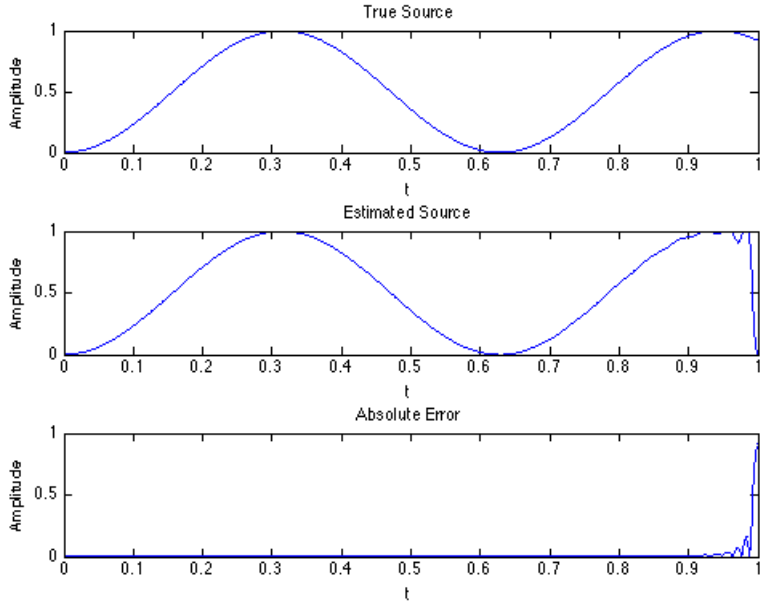

Figure 2: The 2D problem with no noise and one sensor. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom).

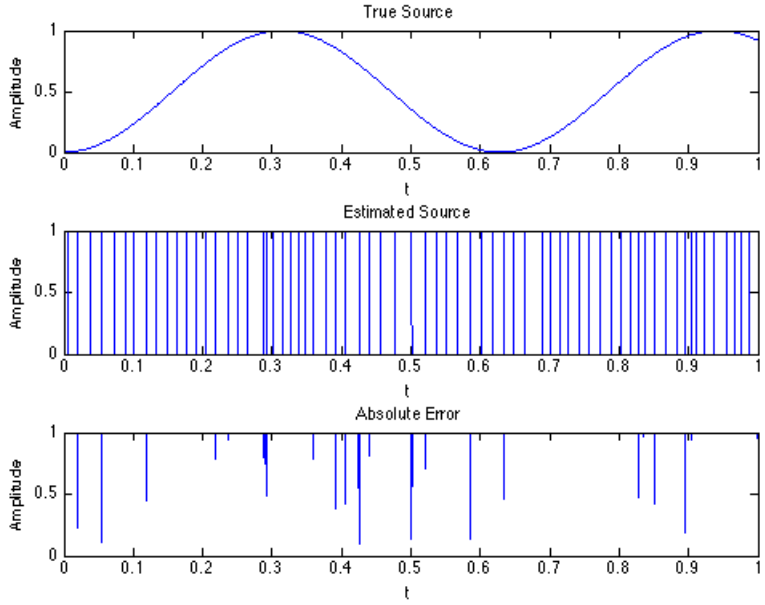

Figure 3: The 2D problem with noise and one sensor. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom).

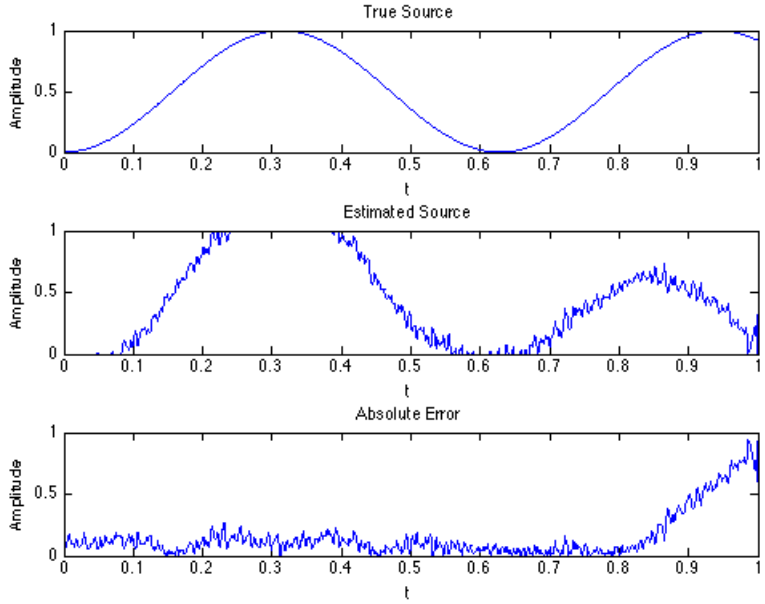

Figure 4: The 2D problem with noise, one sensor, and regularization with α = 1E-3. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom).

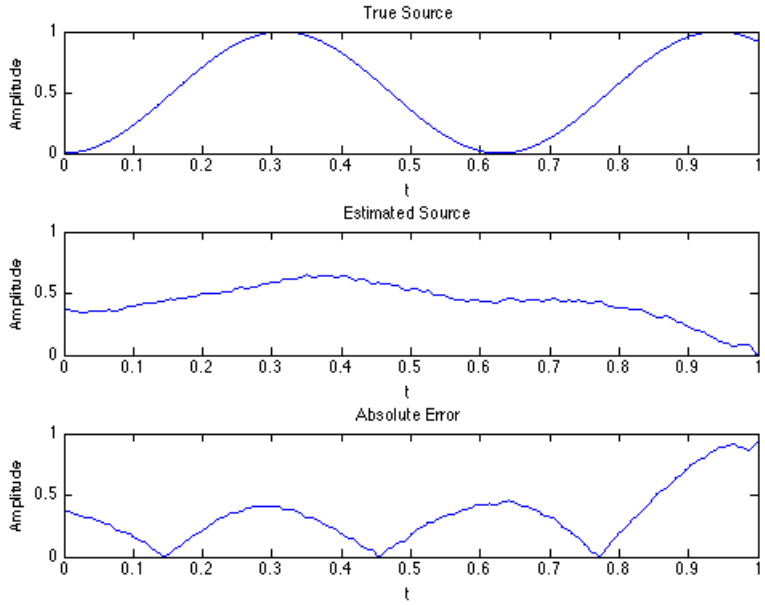

Figure 5: The 2D problem with noise, one sensor, and regularization with α = 1E-2. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom).

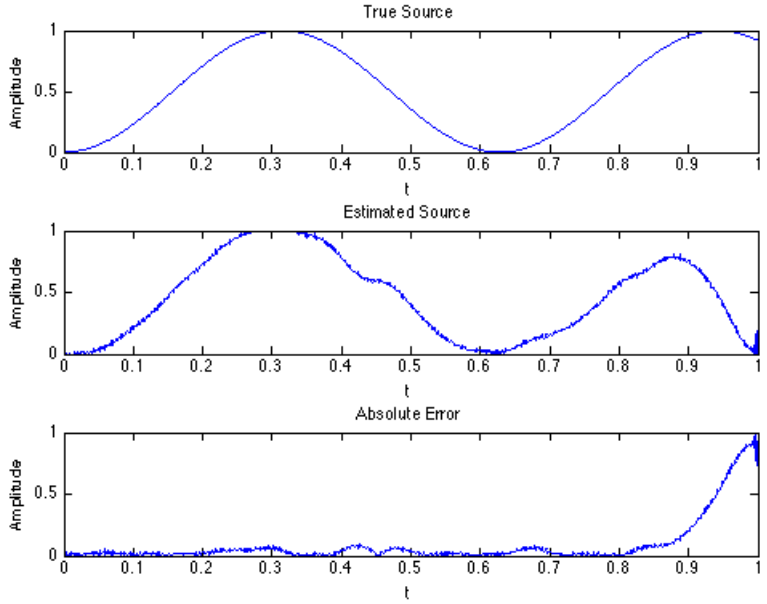

Figure 6: The 2D problem with noise, four sensors, and regularization. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom).

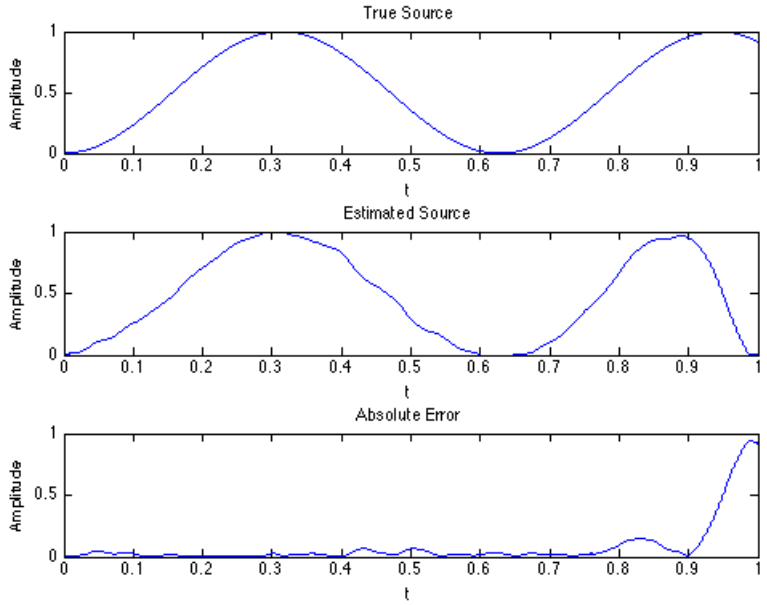

Figure 7: The 2D problem with noise, four sensors, regularization, and temperature samples every 0.1 milliseconds. The figure shows the actual heat source function (top), estimated heat source function (middle), and absolute error between the two (bottom).