A Brief Introduction to Parameter Estimation

David Love, supervised by Dr. Juan Restrepo

Graduate Interdisciplinary Program in Applied Mathematics, University of Arizona

Spring 2008 First Year Term Paper Workshop

A Sample Problem

Parameter estimation is a subset of the study of inverse problems, where we are given the solution and we must deduce the problem. An example

A Sample Problem

Parameter estimation is a subset of the study of inverse problems, where we are given the solution and we must deduce the problem. An example

- Given the answer:

A Sample Problem

Parameter estimation is a subset of the study of inverse problems, where we are given the solution and we must deduce the problem. An example

- Given the answer: 42

A Sample Problem

Parameter estimation is a subset of the study of inverse problems, where we are given the solution and we must deduce the problem. An example

- Given the answer: 42
- What is the question?

Why Estimate Parameters?

- Most physical models contain one or more parameters that cannot be derived mathematically.

Why Estimate Parameters?

- Most physical models contain one or more parameters that cannot be derived mathematically.
- Sometimes these parameters can be found by a simple measurement, e.g., the mass of some reasonably sized object...

Why Estimate Parameters?

- Most physical models contain one or more parameters that cannot be derived mathematically.
- Sometimes these parameters can be found by a simple measurement, e.g., the mass of some reasonably sized object...
- but sometimes they can't.

Why Estimate Parameters?

- Most physical models contain one or more parameters that cannot be derived mathematically.
- Sometimes these parameters can be found by a simple measurement, e.g., the mass of some reasonably sized object...
- but sometimes they can't.
- We need a reliable way to infer these parameters from measured data, taking into account the complications of measurement in the real world.

Parameter System

Abstract parameter system given by the linear system

$$A(c)u = f$$

where A represents the model, as a function of some parameters c, f is a "forcing function" which is known and u is the model prediction based on the current parameters.

Measurements

We are trying to fit the model to some y, a set of (potentially noisy) measured data. The measured data is represented as

$$y = u + \eta$$

where η is the noise inherent in the measurement process. Ideally, η represents a sample of white (Gaussian) noise with zero mean.

Some abstract spaces, assumed to be Hilbert spaces: $\mathcal{Y}$ is the observation space, $\mathcal{U}$ is the state space and $\mathcal{Q}$ is the parameter space.

Objective Function

We must have a way to measure the success of the parameter fitting algorithm, so we need an objective function to be minimized. Generally a regularized least-squares functional is used, given by

$$u^* = \min_{c \in \mathcal{Q}} \left|\left| A(c)^{-1} f - y \right|\right|_2^2 + \gamma J_{reg}(c)$$

where $J_{reg}(c)$ is used to incorporate *a-priori* information into the optimization system (for example, if c must be smooth), and γ is a weight.

Line Search

This project deals with a class of optimization techniques known collectively as the "line search." In a line search optimization you first begin by choosing a direction in which to search for a minima of the function, reducing a multi-dimensional problem to a single-dimensional one.

Then one finds the minima along this line and repeats the process again. Algorithmically, the iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k$$

where x_k is the current guess, p_k is the search direction and α is the step size.

Choice of Search Direction

The first step in a line search is to choose the search direction. In most optimization routines, the search direction is chosen to be a descent direction, i.e.,

$$\nabla f(x_k)^T p_k < 0$$

where $f(x)$ is the functional to be minimized.

Line search methods in this talk are of the form $p_k = -B_k^{-1}\nabla f_k$, where B_k^{-1} is some symmetric, nonsingular matrix. We can ensure that p_k is a descent direction if B_k^{-1} is positive definite, i.e., if $\nabla f_k^T B_k^{-1}\nabla f_k > 0$.

Choice of Step Size

The Wolfe Conditions ensure that the optimization method makes sufficient progress at each iteration. First, one wants to ensure that each step decreases the functional value enough

Choice of Step Size

The Wolfe Conditions ensure that the optimization method makes sufficient progress at each iteration. First, one wants to ensure that each step decreases the functional value enough

Sufficient Decrease Condition

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$$

Choice of Step Size

The Wolfe Conditions ensure that the optimization method makes sufficient progress at each iteration. First, one wants to ensure that each step decreases the functional value enough

Sufficient Decrease Condition

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$$

Second, one does not want to take step sizes too small

Choice of Step Size

The Wolfe Conditions ensure that the optimization method makes sufficient progress at each iteration. First, one wants to ensure that each step decreases the functional value enough

Sufficient Decrease Condition

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$$

Second, one does not want to take step sizes too small

Curvature Condition

$$\nabla f(x_k + \alpha p_k)^T p_k \geq c_2 \nabla f_k^T p_k$$

Parameters c_1 and c_2 are constants. Typical values are $c_1 = 1 \times 10^{-4}$ and $c_2 = 0.9$.

Steepest Descent Method

Perhaps the most obvious idea: take $B_k^{-1} \equiv I$, then $p_k = -\nabla f_k$. In the ideal case, when the objective function is quadratic and and line search is exact, the Steepest Descent method provides quadratic convergence.

In more general cases, Steepest Descent can converge very slowly. This is described by

$$f(x_{k+1}) - f(x^*) \leq r^2 \left(f(x_k) - f(x^*)\right)$$

where r is a scalar in the open interval

$$r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_2 + \lambda_1}, 1\right)$$

and $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of the exact Hessian at x^*

Newton's Method for Root Finding

We all know Newton's Method for finding the roots of a function, given by iterations (in one-dimension)

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Newton's Method for Optimization

This method is modified by simply acknowledging that, in searching for extrema of f we are looking for the roots of f'

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

The obvious extension to higher-dimensions

Newton's Method

$$x_{k+1} = x_k - \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

Newton's Method for Optimization has the same pros and cons as Newton's Method for Root Finding. However, not all extrema are created equal: Newton's Method will find maxima just as easily as minima.

Quasi-Newton Methods

Sometimes calculating the analytic Hessian of a function is impossible or expensive. Quasi-Newton methods are designed to circumvent this problem by approximating the Hessian. The most famous Quasi-Newton method is the BFGS method, named after Broyden, Fletcher, Goldfarb and Shanno.

The BFGS method uses information gained by the previous iteration to update the estimate for the Hessian (actually, the inverse Hessian), by enforcing that the approximation match the gradient at both the current and previous point.

BFGS Inverse Hessian Update

Inverse Hessian Update

$$B_{k+1}^{-1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k} \right) B_k^{-1} \left(I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \frac{s_k s_k^T}{y_k^T s_k}.$$

where $y_k = \nabla f_{k+1} - \nabla f_k$ and $s_k = x_{k+1} - x_k$.

One can get the iteration relation for the Hessian itself by use of the Sherman-Morrison-Woodbury formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

A One-Dimensional Problem

Take the one-dimensional diffusion equation

$$\begin{aligned} u_t(x,t) &= cu_{xx}(x,t) \\ u(0,t) &= u(1,t) \\ u(x,0) &= u_0(x) \end{aligned}$$

where c is some positive constant. The initial distribution $u_0(x)$ is known and the final distribution $u(x,t_f)$ is measured. We want to estimate c.

PDE Implementation

Break up unit interval into equidistant nodes

$$0 = x_1 < x_2 < x_3 < \cdots < x_N < x_{N+1} = 1.$$

The diffusion term is approximated by a center difference

$$u_{xx}(x_j, t) \approx \frac{u(x_{j+1}, t) - 2u(x_j, t) + u(x_{j-1}, t)}{\Delta x^2}.$$

Then the PDE is converted into a system of coupled ODEs, given by the system $u_t = cLu$.

PDE Implementation

L is the discrete Laplacian with periodic boundary conditions

$$L = \begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & 0 & \dots & 0 & 1 & -2 \end{pmatrix}.$$

Implicit Euler is used for the time derivative, $\frac{1}{\Delta t}(u^{n+1} - u^n) = \frac{c}{\Delta x^2} L u^{n+1}$, which simplifies to

$$A^n u^n = u^0$$

where $u^n = u(n\Delta t) = (u(x_0, n\Delta t), \dots, u(x_N, n\Delta t))^T$, and A is the $N \times N$ matrix $I - c\frac{\Delta t}{\Delta x^2} L$.

Optimization Implementation

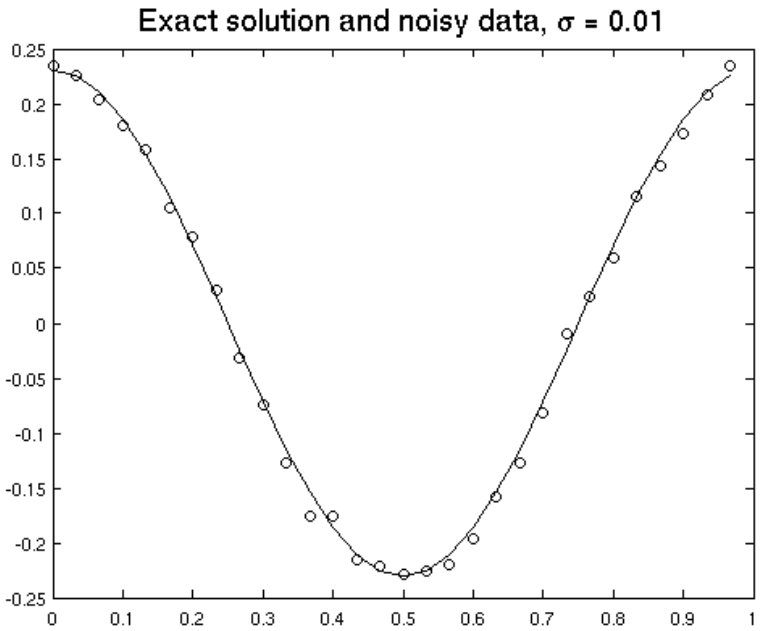
The cost function becomes

$$\min_{c>0} \|F(c) - d\|_2^2$$

where $F(c) = A(c)^{-n}u^0$ and d is the measured, potentially noisy data.

Optimization was a simple Newton's Method.

A Look at the Measurements



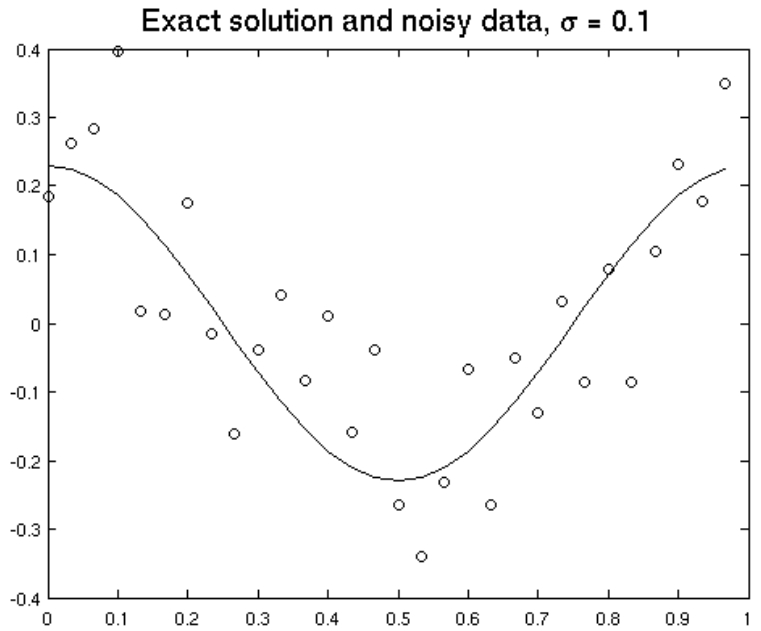


Figure: True solution to the simple initial value problem $u_t = cu_{xx}$ across a bar of unit length with boundary conditions $u(0,t) = u(1,t)$ and constant c. Gaussian noise is added to the true solution to generate the input data which the parameter fitting algorithm is to work with. The noise has zero mean in both cases, with standard deviation 0.01 (left) and 0.1 (right).

A Look at the Optimization Surface

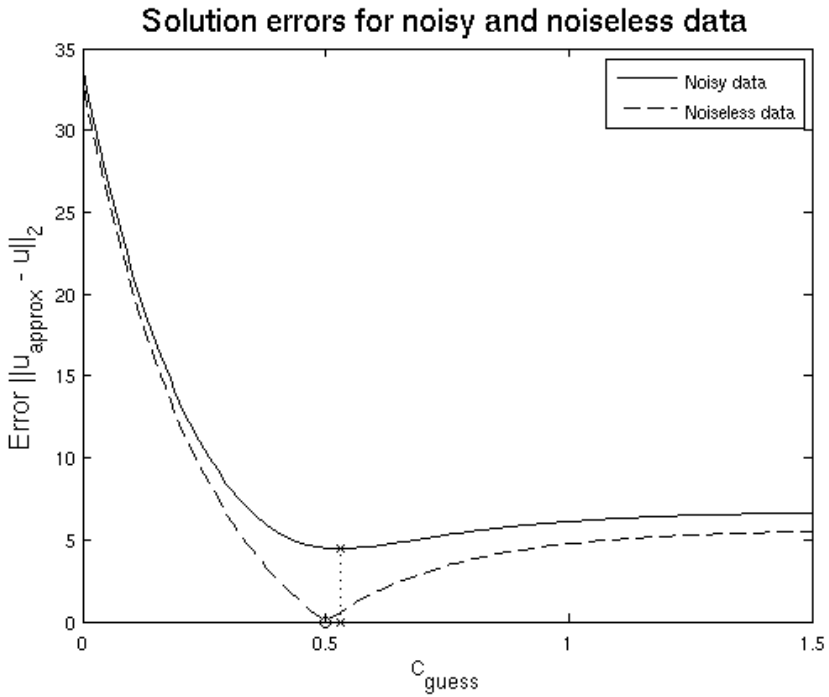


Figure: 2-norm errors in the solution to the initial value problem as a function of the initial guess c_{guess}, with $c_{true} = 0.5$. The curve has two areas of interest: a convex portion to the left of the minimizer and a concave portion to the right. When using a (quasi-)Newton method for optimization the function iterates must stay in the convex portion, or the algorithm will attempt to find a maxima.

Diffusion Constant Error

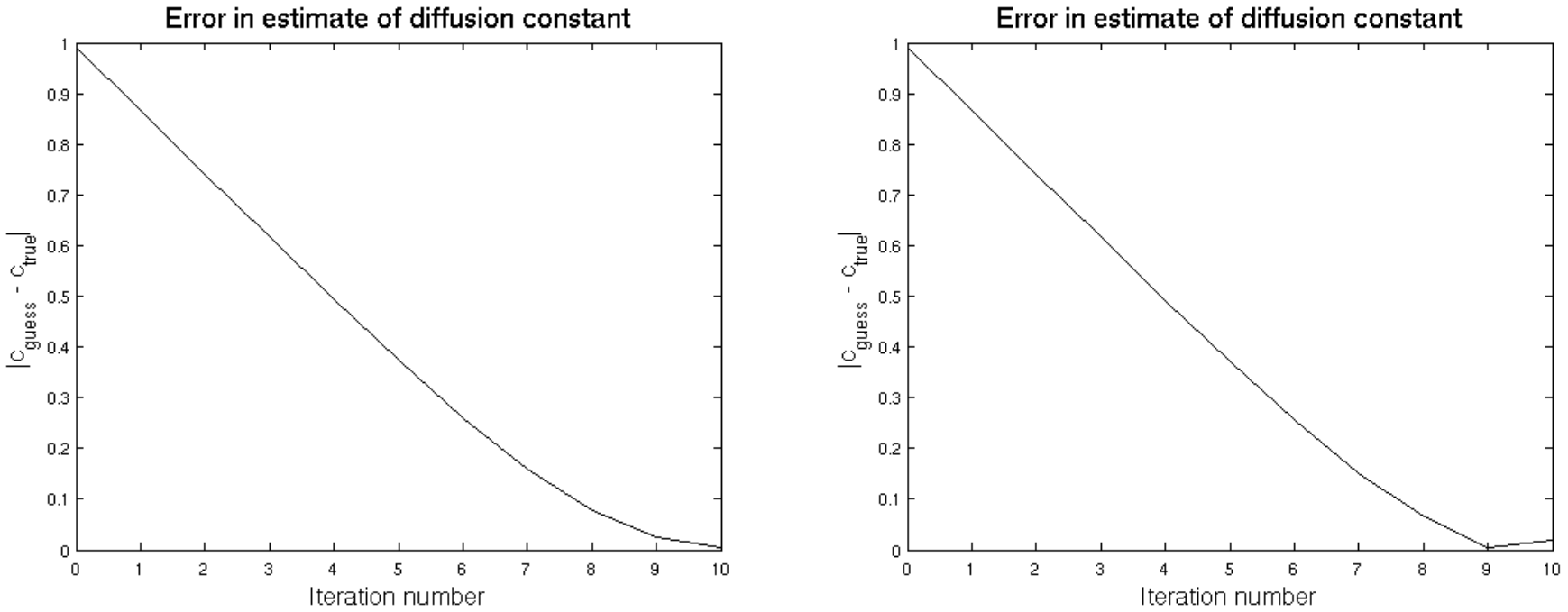
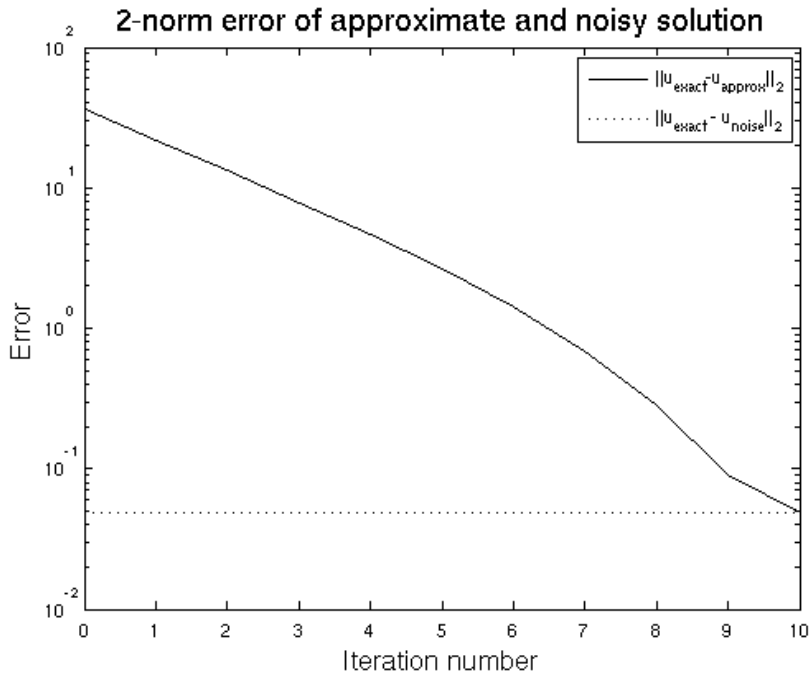
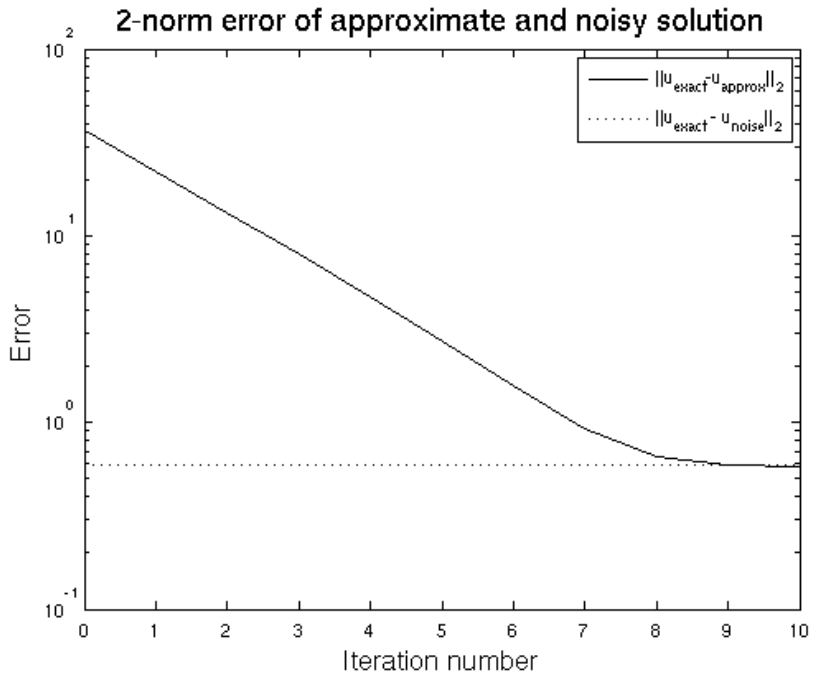


Figure: Convergence plots of two examples of the simple initial value problem $u_t = cu_{xx}$ with noisy data. These solutions were taken for $\sigma = 0.01$ (left) and $\sigma = 0.1$ (right). We note that the convergence is only linear instead of quadratic that one generally expects with Newton's method.

Error of Solution





A Space-Varying Parameter Problem

The previous diffusion equation is modified so that the diffusion parameter c is allowed to vary with space

$$\begin{aligned} u_t(x,t) &= c(x)u_{xx}(x,t) \\ u(0,t) &= u(1,t) \\ u(x,0) &= u_0(x) \end{aligned}$$

Implementation Changes

Very little changes in the implementation of the PDE system. The Laplacian matrix has changed

$$L = \begin{pmatrix} -2c_1 & c_1 & 0 & 0 & \dots & c_1 \\ c_2 & -2c_2 & c_2 & 0 & \dots & 0 \\ 0 & c_3 & -2c_3 & c_3 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ c_N & 0 & \dots & 0 & c_N & -2c_N \end{pmatrix}$$

For the remainder of the examples, it is assumed that $c(x)$ is a piecewise constant function.

Optimization Implementation

- Having both concave and convex regions is more significant in the multi-dimensional case.

Optimization Implementation

- Having both concave and convex regions is more significant in the multi-dimensional case.
- I have chosen to use a hybrid Steepest Descent–BFGS method to overcome this.

Optimization Implementation

- Having both concave and convex regions is more significant in the multi-dimensional case.
- I have chosen to use a hybrid Steepest Descent–BFGS method to overcome this.
- The Quasi-Newton method will do the bulk of the work

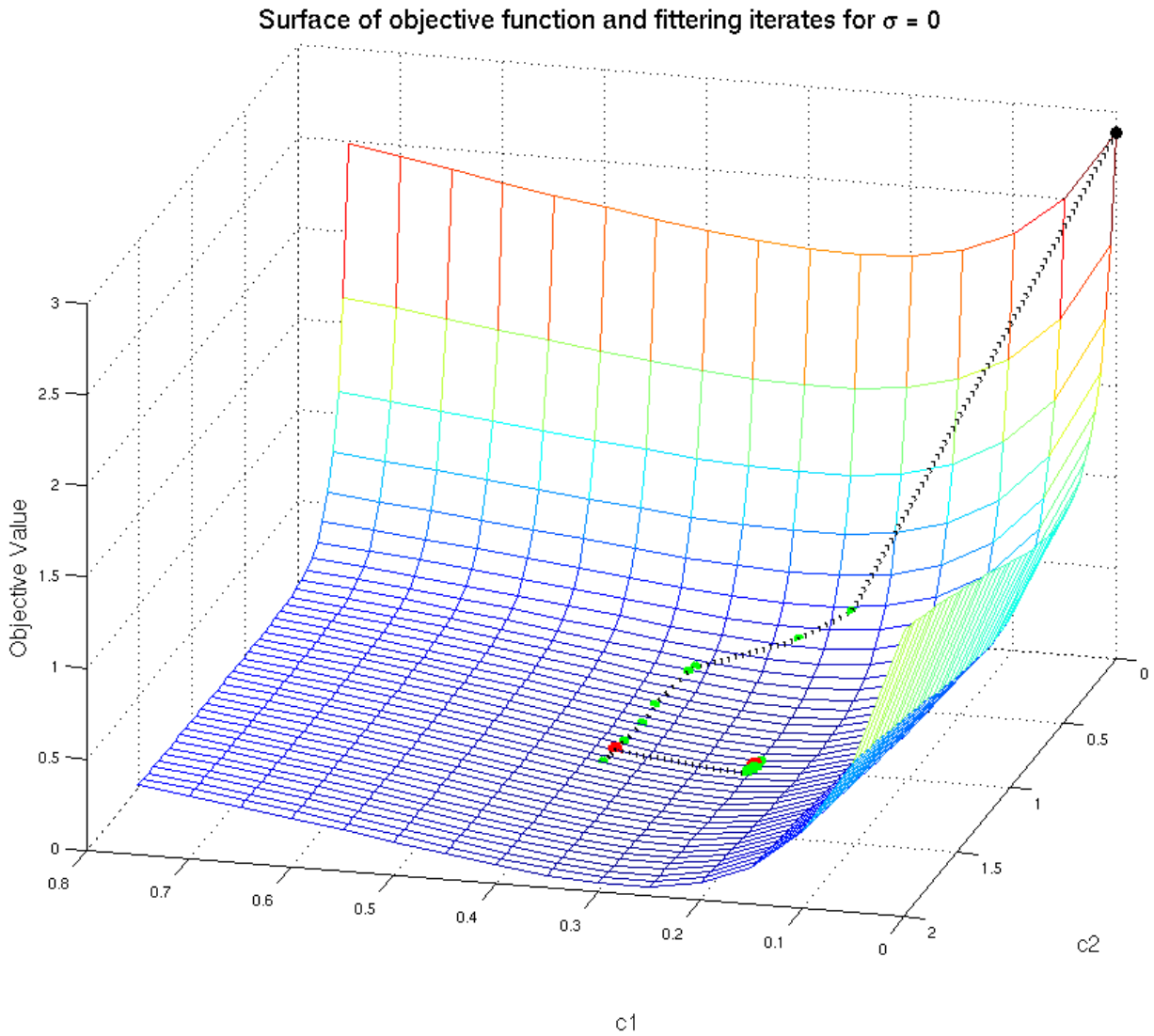
Optimization Implementation

- Having both concave and convex regions is more significant in the multi-dimensional case.
- I have chosen to use a hybrid Steepest Descent–BFGS method to overcome this.
- The Quasi-Newton method will do the bulk of the work
- Whenever a BFGS search direction is not a descent direction a step of Steepest Descent will be used

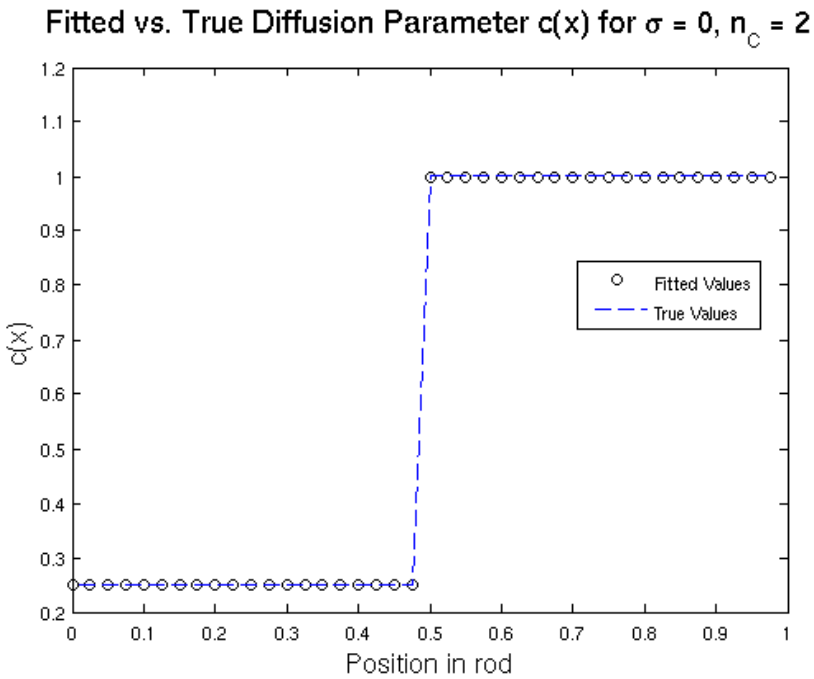
Optimization Implementation

- Having both concave and convex regions is more significant in the multi-dimensional case.
- I have chosen to use a hybrid Steepest Descent–BFGS method to overcome this.
- The Quasi-Newton method will do the bulk of the work
- Whenever a BFGS search direction is not a descent direction a step of Steepest Descent will be used
- Hessian approximates will be updated as normal when the Steepest Descent method is used.

Rainbow Landscape in Two Dimensions



Comparison of True vs. Fitted Parameters and Function Values



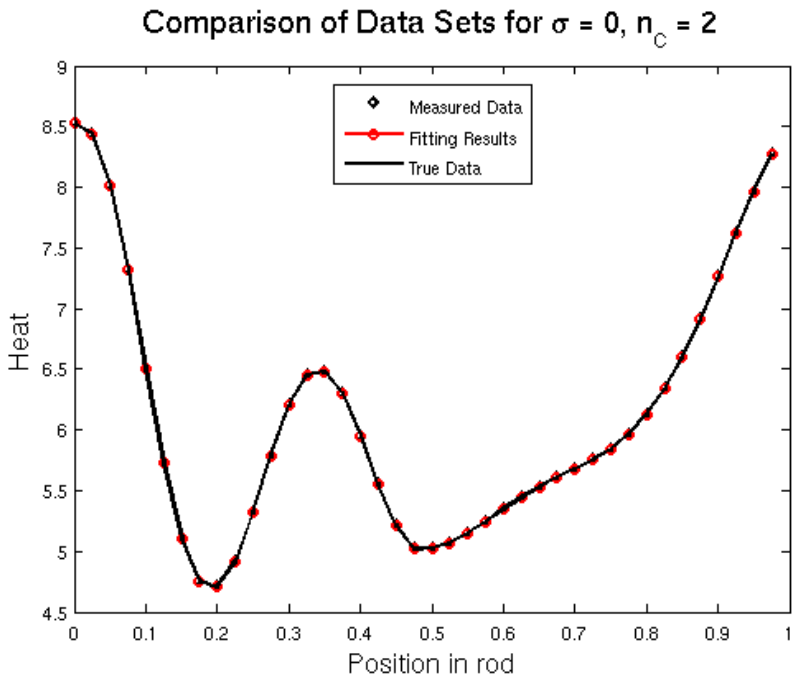


Figure: Comparisons of the true and estimated values for the diffusion parameters (left) and the measured and calculated solutions to the PDE (right). The algorithm has little trouble picking out the true parameters in this low-dimensional, noiseless system.

Error as the Optimization Progresses

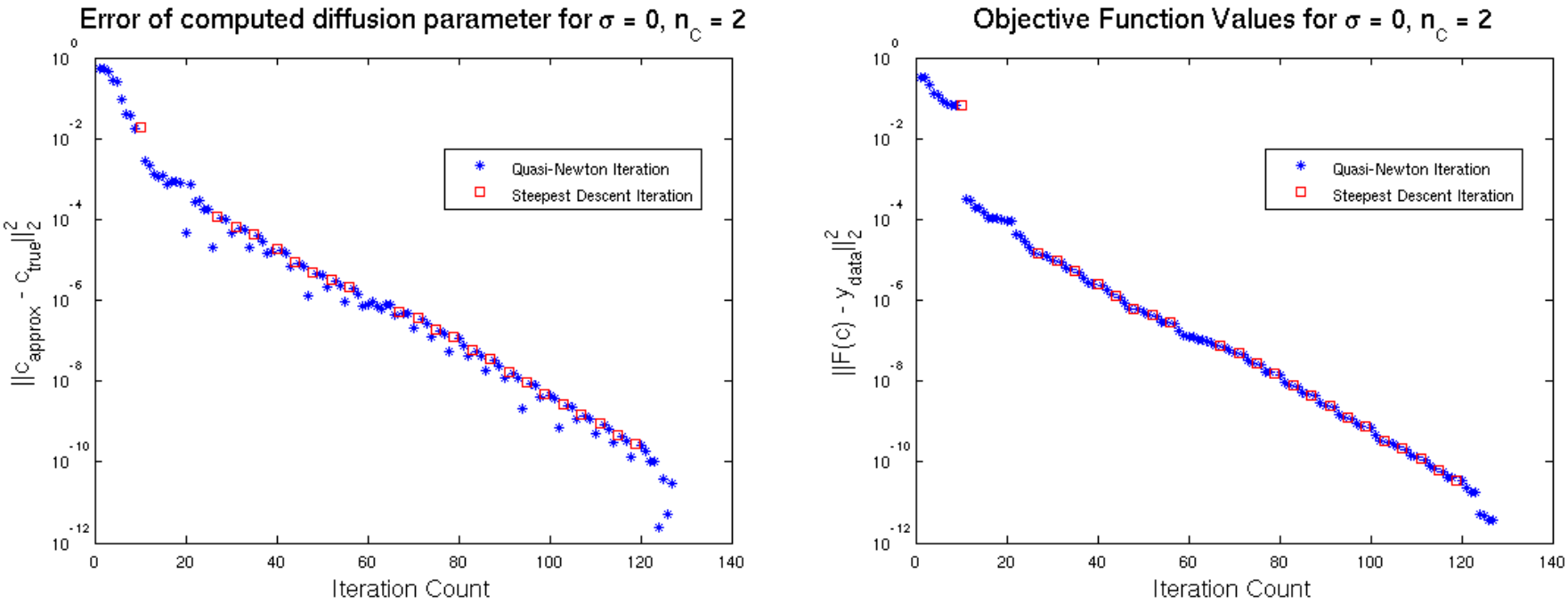


Figure: Error of the parameter estimates (left) and calculated function values (right) over the length of the optimization. The error of the parameter estimate decreases about like $e^{-0.06k}$ for the bulk of the optimization loop.

A bit more fun...

All well and good, but that problem was too easy...

We'll have a bit more fun...

A bit more fun...

All well and good, but that problem was too easy...

We'll have a bit more fun...

- Increase the number of parameters to 8

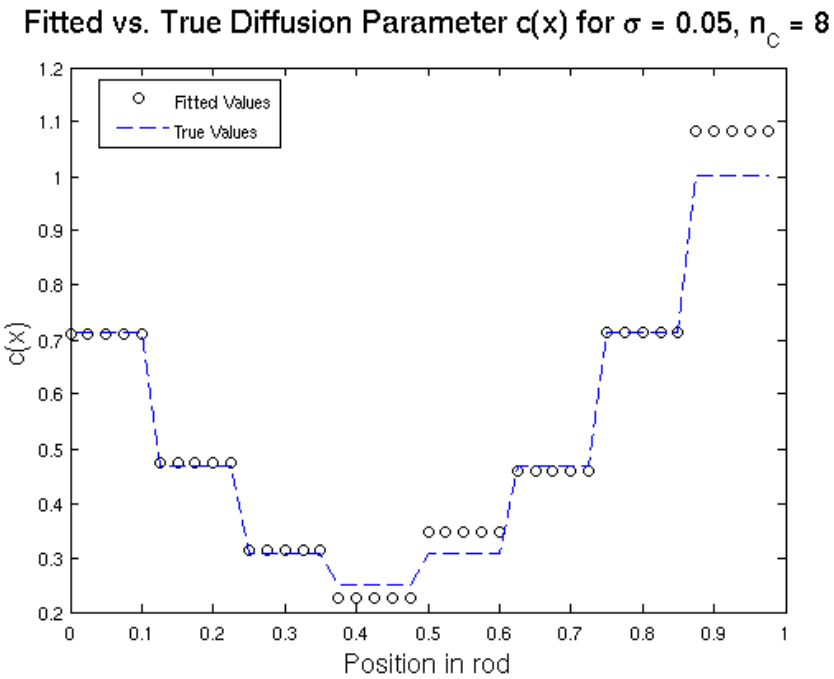
A bit more fun...

All well and good, but that problem was too easy...

We'll have a bit more fun...

- Increase the number of parameters to 8
- Allow for some white noise of standard deviation $\sigma = 0.05$

Comparison of True vs. Fitted Parameters and Function Values



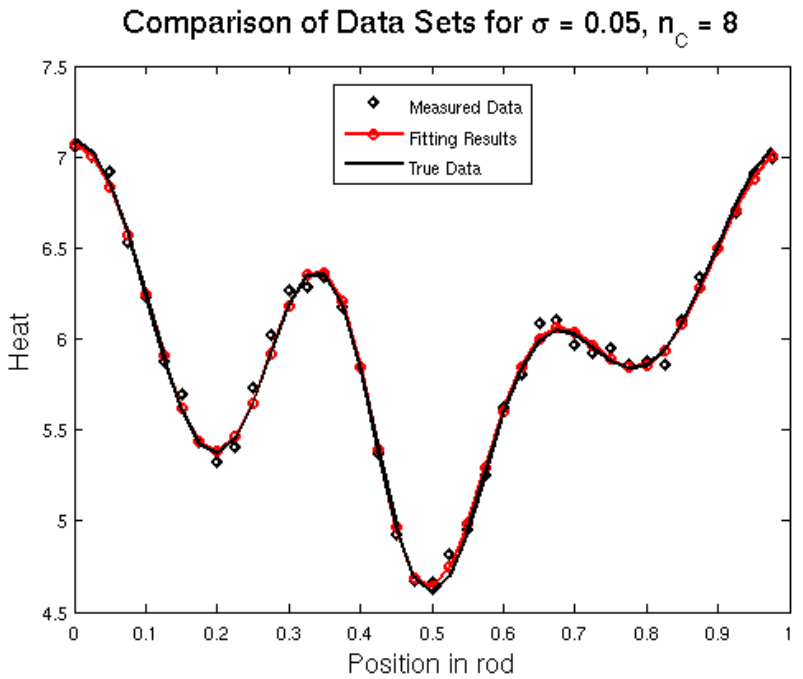


Figure: Comparisons of the true and estimated values for the diffusion parameters (left) and the measured and calculated solutions to the PDE (right). The algorithm still picks out the overall shape of $c(x)$, but does not get the values exactly.

Error as the Optimization Progresses

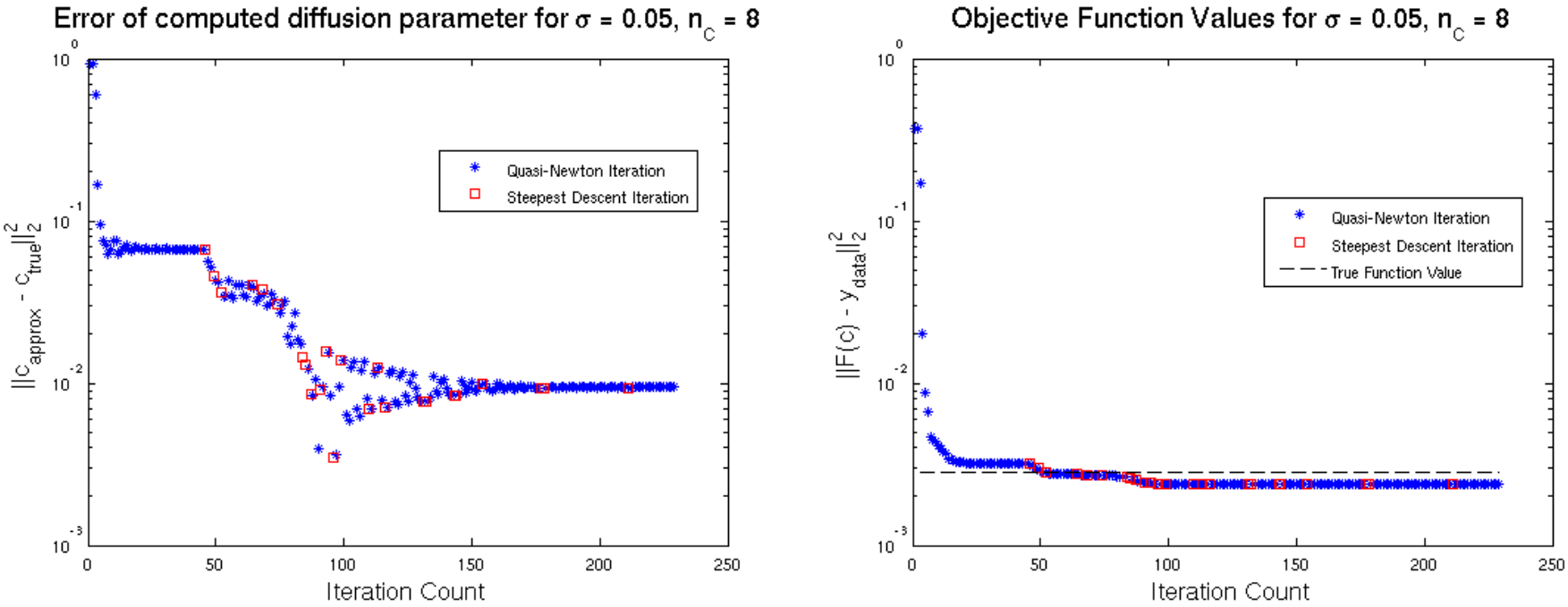


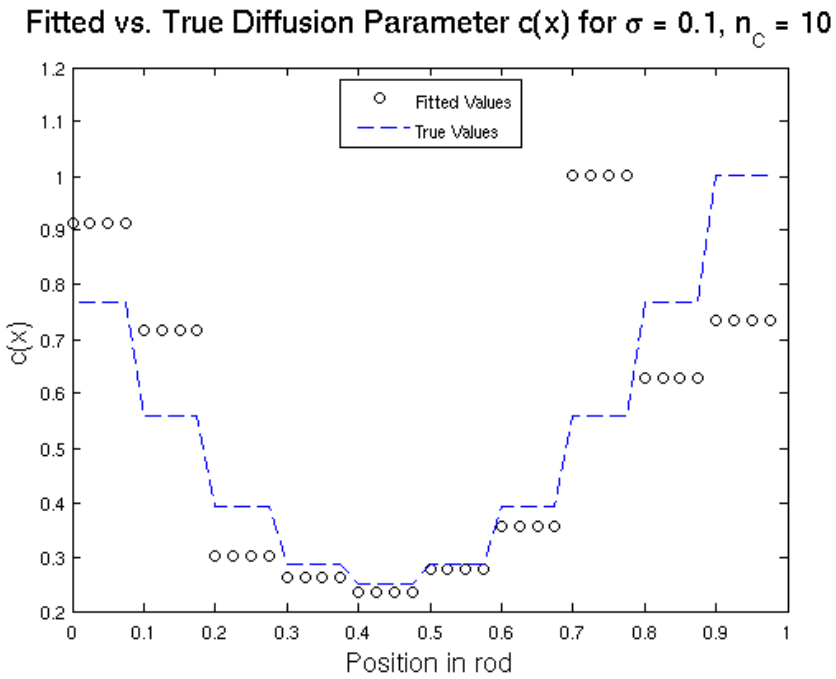
Figure: Error of the parameter estimates (left) and calculated function values (right) over the length of the optimization. The optimization reaches several walls, approximately at the level of the noise, which it has difficulty breaking through.

One More Example (of Failure?)

This time with feeling

- Increase the number of parameters to 10
- Double the noise level

Comparison of True vs. Fitted Parameters and Function Values



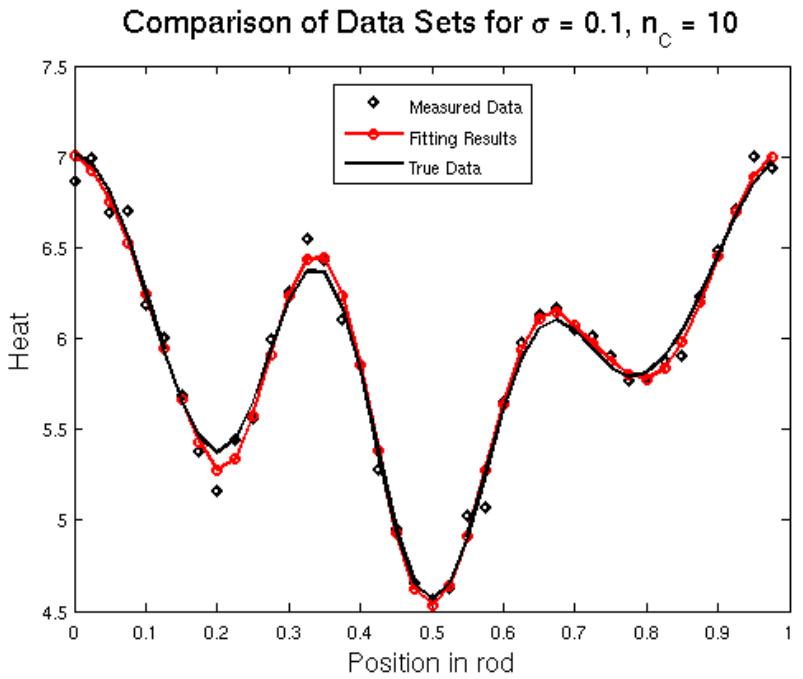


Figure: Comparisons of the true and estimated values for the diffusion parameters (left) and the measured and calculated solutions to the PDE (right). The noise in the measurements pull the algorithm away from the true solution, towards another solution.

Error as the Optimization Progresses

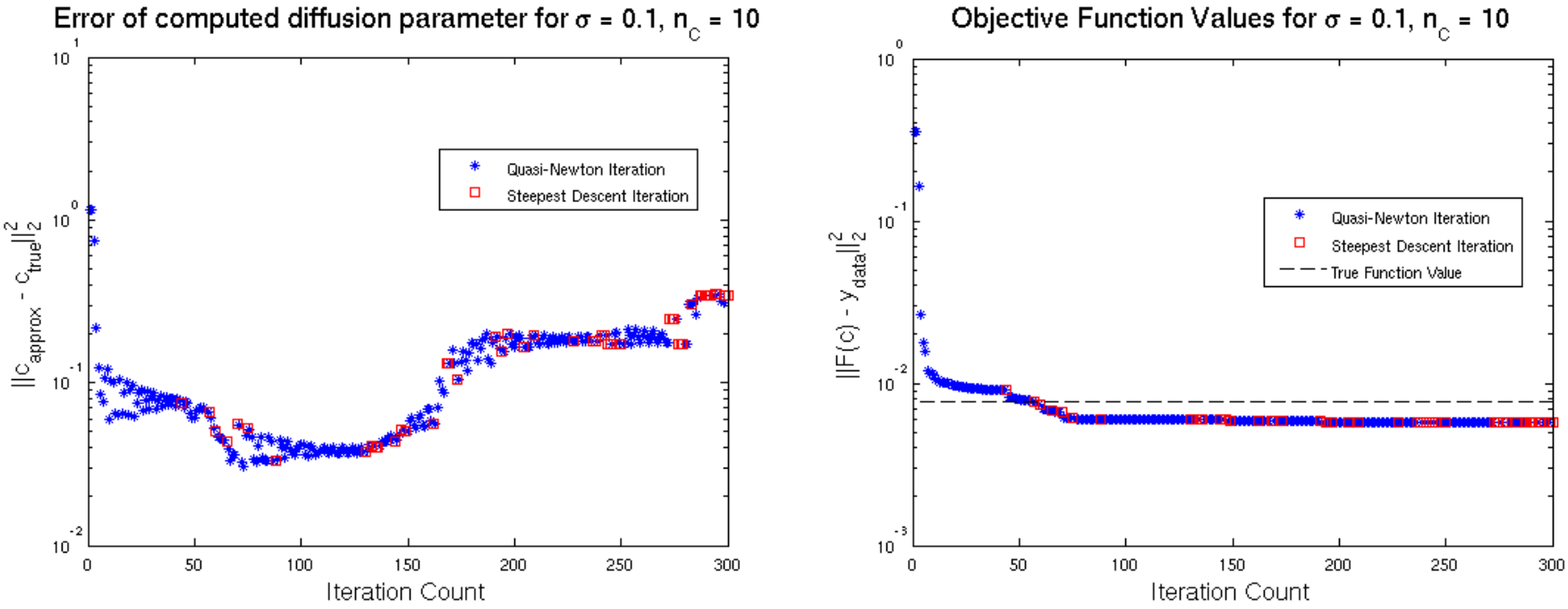


Figure: Error of the parameter estimates (left) and calculated function values (right) over the length of the optimization. After some time the algorithm actually moves away from the true solution, but the error decreases the entire time. Optimization was stopped by hitting the maximum number of iterations.

Conclusion

- In the noiseless case, the hybrid algorithm converges spectrally to the true solution.
- Noise creates an effective barrier to optimization, which takes a long time to overcome.

Acknowledgments

The presenter would like to thank Dr. Juan Restrepo for his guidance in this project.

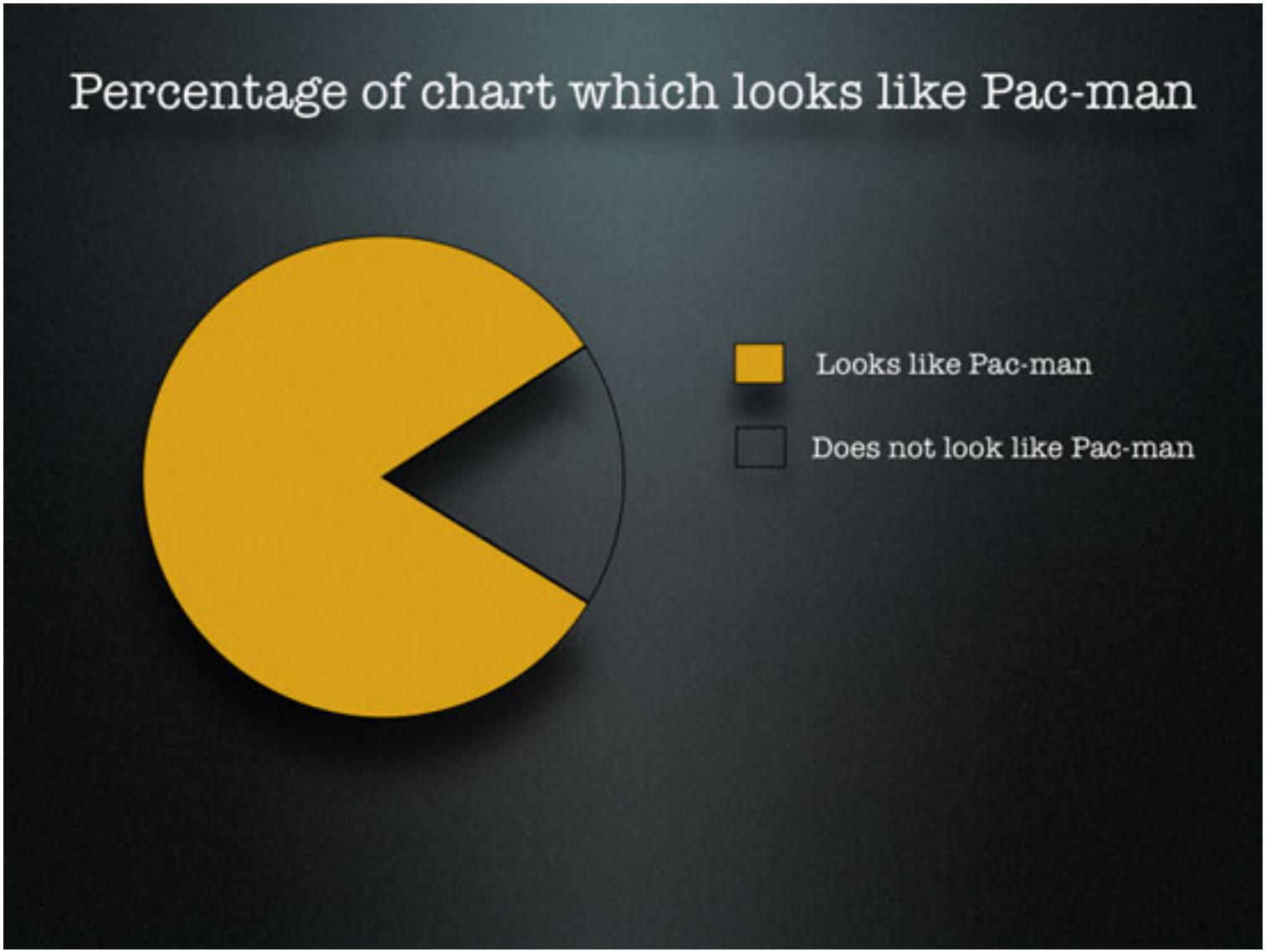
References

[1] Dimitri P. Bertsekas. Nonlinear Programming. Athena Scientific, Belmont, Mass., second edition, 1999.

[2] Jorge Nocedal and Stephen J. Wright. Numerical Optimization. Springer, New York, NY, second edition, 2006.

[3] Curtis R. Vogel. Computational Method for Inverse Problem. SIAM, Philadelphia, PA, 2002.

Questions?


Percentage of chart which looks like Pac-man
Looks like Pac-man
Does not look like Pac-man