

# CAAM 454 Project: An Inverse Problem

Instructor Yin Zhang

April 17, 2007, revised 2008

## 1 The Model

Consider a discrete linear model (the forward problem)

$$A(s)u = b, \tag{1}$$

where  $A \in \mathfrak{R}^{n \times n}$  contains unknown parameter  $s \in \mathfrak{R}^m$  and is nonsingular for reasonable  $s$  values. Some “true” parameter value  $s^*$  exists for the model.

For any given right-hand side  $b$ , we can partially observe

$$u(s^*, b) = A(s^*)^{-1}b.$$

That is, for some index set  $B \subset \{1, 2, \dots, n\}$ , we can approximately observe  $u_i(s^*, b)$  for all  $i \in B$ ; or simply, the sub-vector  $u_B(s^*, b) \in \mathfrak{R}^{|B|}$ . Let  $n_B = |B|$ , the cardinality of  $B$ .

The problem of finding  $s^*$  is called an inverse problem. To find  $s^*$ , we do a number of experiments by giving  $b^e \in \mathfrak{R}^n$  and observing the corresponding

$$c^e \approx u_B(s^*, b^e) \in \mathfrak{R}^{n_B}, \quad e = 1, 2, \dots, E. \tag{2}$$

Then we use least-squares fitting to hopefully find an approximation to  $s^*$

by solving

$$\min_s \phi(s) := \frac{1}{2} \sum_{e=1}^E \|u_B(s, b^e) - c^e\|_2^2. \quad (3)$$

However, this is generally an ill-posed problem that may permit different solutions fitting the observed data almost equally well.

## 2 The Gradient and Jacobian

You will need to calculate the gradient of  $\phi(s)$ . Derive detailed formulas for  $\nabla\phi(s)$ , and an efficient algorithm to compute the gradient of  $\phi(s)$  at any given  $s \in \mathfrak{R}^m$  value.

Since all the terms in the sum of  $\phi(s)$  has the same form, for simplicity let us assume  $E = 1$  and drop the superscript  $e$ . In this case, we consider

$$\phi(s) := \frac{1}{2} \|u_B(s) - c\|_2^2 := \frac{1}{2} r_B(s)^T r_B(s). \quad (4)$$

we can derive, as demonstrated in class, that

$$[\nabla\phi(s)]_i = -p(s)^T A'_i(s)u(s), \quad i = 1, 2, \dots, m, \quad (5)$$

where  $A'_i(s) = \partial A(s)/\partial s_i$ , and  $p(s)$  satisfies the linear system

$$A^T(s)p = \hat{r}(s),$$

where  $\hat{r}(s) = [r_B(s); 0]^T$  with an appropriate arrangement (i.e., the components are zeros for indices not in  $B$ ).

In general, when  $E > 1$ ,  $\nabla\phi(s)$  should be a sum of  $E$  terms of the form on the right-hand side of (5) for each of its components.

To use Gauss-Newton or Levenberg-Marquardt method, we will need the Jacobian of the residual  $r_B(s)$ , where  $r(s)$ . Recall that the gradient can also be written as

$$\nabla\phi(s) = J_B(s)^T r_B(s).$$

We can derive a formula:

$$J_B = - (A^{-1}[A'_1 u \ A'_2 u \ \cdots \ A'_m u])_B.$$

How to really compute the Jacobian efficiently is problem-dependent.

### 3 An Inverse Problem

Consider the two-point boundary value problem (BVP) of ODE:

$$u''(x) - \alpha(x)u(x) = 0, \quad x \in (0, 1), \quad u(0) = u^0, \quad u(1) = u^1, \quad (6)$$

where the unknown parameter is  $\alpha(x) > 0$ . We assume that we know that  $\alpha(x)$  is linear in  $[0, 1]$ . For any given boundary values, we can observe (measure) the corresponding solution  $u(x)$  at a few given locations in  $(0, 1)$ .

We will discretize the above BVP on a uniform grid by a finite-difference scheme, and obtain a model in the form of (1). Our task is to estimate  $\alpha(x)$  using the least-squares model (3).

This simple problem is well-posed and easy to solve; hence no regularization or other stabilization techniques is necessary.

### 4 Assignment and Requirements

Implement gradient descent and Gauss Newton methods and write well documented codes to solve problem (3) associated with the BVP (6), and then report what you did and what are your findings.

Please see the course website for further instructions and specifications on required tests.

Write a formal, typeset, report that (i) gives derivation of formulas and the algorithms you use in this project with clearly defined notations (you will

need some); (ii) presents, describes and interprets your results. The main body of your report (excluding plots, if any) should not exceed 4 pages.

Submit your report and your codes, as well as other accompanying materials if any, through Owl-Space.

Your project will be graded based on (i) the soundness of your approach, (ii) the quality and efficiency of your codes, and (iii) the clarity of your report.

## 5 Instructor's Approach

It is perhaps helpful to some of you to describe what I did. I wrote a Matlab function to evaluate the residual  $r(s)$  and its Jacobian  $J(s)$  (actually  $r_B$  and  $J_B$  in previous notation), as shown below,

```
function [r, J] = zinvprJ(s,n,bvsE,cE,B)
% Evaluate residual and its Jacobian
%   Input:
%       s - parameter represent linear functions (2 x 1)
%       n - number of interior nodes in a uniform grid
%       bvsE - boundary values used in experiments (2 x E)
%       cE - observed data in experiments (|B| x E)
%       B - indices for observation locations
%   Output:
%       r - residual at s
%       J - Jacobian of r at s
```

In the function `zsolver`, I implemented a gradient descent and a Gauss-Newton (GN) method, both with a back-tracking line search, to solve the

problem (see the test script for its interface).

I also wrote a function, `function [A,b] = zgetAb(s,bvs,n)`, to compute the matrix  $A(s)$  at any given  $s$ , and the right-hand side  $b$  associated with the boundary values given in `bvs`. This function is used in the test script to generate data.