Blocky models via the L1/L2 hybrid norm

Jon Claerbout

ABSTRACT

This paper seeks to define robust, efficient solvers of regressions of $L1$ nature with two goals: (1) straightforward parameterization, and (2) "blocky" solutions. It uses an $L1/L2$ hybrid norm characterized by a residual R_d of transition between $L1$ and $L2$ for data fitting and another R_m for model styling. Both the steepest descent and conjugate direction methods are included. The 1-D blind deconvolution problem is formulated in a manner intended to lead to both a blocky impedance function and a source waveform. No results are given.

INTRODUCTION

I've seen many applications improved when least-squares ($L2$) model fitting was changed to least absolute values ($L1$). I've never seen the reverse. Never-the-less we always return to $L2$ because the solving method is easier and faster. It does not require us to specify parameters of numerical analysis that are unclear how to specify.

Another reason to re-investigate $L1$ is its natural ability to estimate blocky models. Sedimentary sections tend to fluctuate randomly, but sometimes there is a homogeneous material continuing for some distance. A function with such homogeneous regions is called "blocky". The derivative of such a function is called "sparse". $L2$ gives huge penalties to large values and minuscule penalties to small ones, hence it never really produces sparse functions and their integrals are never really blocky. If we had an easy, reliable $L1$ solver, we could expect to see many more realistic solutions.

There are reasons to abandon strict $L1$ and revert to an $L1/L2$ hybrid solver. A hybrid solver has a parameter, a threshold, at which $L2$ behavior transits to $L1$. We have good reasons to use two different hybrid solvers, one for the data fitting, the other for the model styling (prior knowledge or regularization). Each requires a threshold of residual, let us call it R_d for the data fitting, and R_m for the model styling. Processes that require parameters are detestible when we have a poor idea of the meaning of the parameters (especially if they relate to numerical analysis), however the meaning of the thresholds R_d and R_m is quite clear. When we look at a shot gather and see about 30% of the area is covered with ground roll, it is clear we would like to choose R_d to be at about the 70th percentile of the fitting residual. As for the model styling, if we'd like to see blocks about 20 points long, we'd like our spikes to average about 20 points apart, so we would like R_m about the 95th percentile allowing 5% of the spikes to be of unlimited size, while the others small.

I was first attracted to strict $L1$ by its potential for blocky models. But then I realized for each nonspike (zero) on the time axis, theory says I would need a "basis equation". That implies an immense number of iterations, so it is unacceptable in imaging applications. With the hybrid solvers, instead of exact zeros we have a large region driven down by the L2 norm and a small L1 region where large spikes are welcomed.

MODEL DERIVATIVES

Here is the usual definition of residual r_i of theoretical data $\sum_j F_{i,j} m_j$ from observed data d_i

$$r_i = (\sum_j F_{i,j} m_j) - d_i \qquad \text{or} \qquad \mathbf{r} = \mathbf{F}\mathbf{m} - \mathbf{d}. \tag{1}$$

Let $C()$ be a convex function ($C'' \geq 0$) of a scalar. The penalty function (or norm of residuals is expressed by

$$N(\mathbf{m}) = \sum_i C(r_i) \tag{2}$$

We denote a column vector $\mathbf{g}$ with components g_i by $\mathbf{g} = \text{vec}(\text{g}_\text{i})$. We soon require the derivative of $C(r)$ at each residual r_i:

$$\mathbf{g} \quad = \quad \text{vec}\left[\frac{\partial \text{C}(\text{r}_\text{i})}{\partial \text{r}_\text{i}}\right] \tag{3}$$

We often update models in the direction of the gradient of the norm of the residual.

$$\Delta\mathbf{m} \quad = \quad \frac{\partial N}{\partial m_k} \quad = \quad \sum_i \frac{\partial C(r_i)}{\partial r_i}\frac{\partial r_i}{\partial m_k} \quad = \quad \sum_i g(r_i) F_{i,k} \quad = \quad \mathbf{F}'\mathbf{g} \tag{4}$$

Define a model update direction by $\Delta\mathbf{m} = \mathbf{F}'\mathbf{g}$. Since $\mathbf{r} = \mathbf{F}\mathbf{m} - \mathbf{d}$, we see the residual update direction will be $\Delta\mathbf{r} = \mathbf{F}\Delta\mathbf{m}$. To find the distance α to move in those directions

$$\mathbf{m} \quad \leftarrow \quad \mathbf{m} + \alpha\Delta\mathbf{m} \tag{5}$$

$$\mathbf{r} \quad \leftarrow \quad \mathbf{r} + \alpha\Delta\mathbf{r} \tag{6}$$

we choose the scalar α to minimize

$$N(\alpha) = \sum_i C(r_i + \alpha\Delta r_i) \tag{7}$$

The sum in equation (7) is a sum of "dishes", shapes between L2 parabolas and L1 V's. The i-th dish is centered on $\alpha = -r_i/\Delta r_i$. It is steep and narrow if Δr_i is large,

and low and flat where Δr_i is small. The positive sum of convex functions is convex. There are no local minima. We can get to the bottom by following the gradient. Next we consider some choices for convex functions. We'll need them, their first and second derivatives.

Some convex functions and their derivatives

LEAST SQUARES:

$$C = r^2/2 \tag{8}$$
$$C' = r \tag{9}$$
$$C'' = 1 \quad \geq 0 \tag{10}$$

L1 NORM:

$$C = |r| \tag{11}$$
$$C' = \text{sgn}(r) \tag{12}$$
$$C'' = 0 \text{ or } \infty \quad \geq 0 \tag{13}$$

HYBRID:

$$C = R^2(\sqrt{1 + r^2/R^2} - 1) \tag{14}$$
$$C' = \frac{r}{\sqrt{1 + r^2/R^2}} \tag{15}$$
$$C'' = \frac{1}{(1 + r^2/R^2)^{3/2}} \quad \geq 0 \tag{16}$$

HUBER:

$$C = \left\{ \begin{array}{ll} |r| - R/2 & \text{if } |r| \geq R \\ r^2/2R & \text{otherwise} \end{array} \right\} \tag{17}$$
$$C' = \left\{ \begin{array}{ll} \text{sgn}(r) & \text{if } |r| \geq R \\ r/R & \text{otherwise} \end{array} \right\} \tag{18}$$
$$C'' = \left\{ \begin{array}{ll} 0 \text{ or } \infty & \text{if } |r| \geq R \\ 1/R & \text{otherwise} \end{array} \right\} \quad \geq 0 \tag{19}$$

I have scaled Hybrid so it naturally approaches the least squares limit as $R \to \infty$. As $R \to 0$, it tends to $C = R|r|$, scaled $L1$.

Because of the erratic behavior of C'' for $L1$ and Huber, and our planned use of second order Taylor series, we will not be using $L1$ and Huber norms here. Also, we should prepare ourselves for danger as HYBRID approaches the $L1$ limit. (I thank Mandy for reminding me of the infinite second derivative and I thank Mohammad and Nader for demonstrating numerical erratic behavior.)

PLANE SEARCH

The most universally used method of solving immense linear regressions such as imaging problems is the Conjugate Gradient (CG) method. It has the remarkable property that in the presence of exact arithmetic, the exact solution is found in a finite number of iterations. A simpler method with the same property is the Conjugate Direction method. It is debatable which has the better numerical roundoff properties, so we generally use the Conjugate Direction method as it is simpler to comprehend. It says not to move along the gradient direction line, but somewhere in the plane of the gradient and the previous step. The best move in that plane requires us to find two scalars, one α to scale the gradient, the other β to scale the previous step. That is all for $L2$ optimization. We proceed here in the same way with other norms and hope for the best.

So here we are, embedded in a giant multivariate regression where we have a bivariate regression (two unknowns). From the multivarate regression we are given three vectors in data space. $\bar{r}_i$, g_i and s_i. You will recognize these as the current residual, the gradient (Δr_i), and the previous step. (The gradient and previous step appearing here have previously been transformed to data space (the conjugate space) by the operator $\mathbf{F}$.) Our next residual will be a perturbation of the old one.

$$r_i \quad = \quad \bar{r}_i \; + \; \alpha g_i \; + \; \beta s_i \tag{20}$$

We seek to minimize by variation of (α, β)

$$N(\alpha, \beta) = \sum_i \; C(\bar{r}_i + \alpha g_i + \beta s_i) \tag{21}$$

Let the coefficients (C_i, C_i', C_i'') refer to a Taylor expansion of $C(r)$ about r_i.

$$N(\alpha, \beta) = \sum_i \; C_i \; + \; (\alpha g_i + \beta s_i) C_i' \; + \; (\alpha g_i + \beta s_i)^2 C_i''/2 \tag{22}$$

We have two unknowns, (α, β) in a quadratic form. We set to zero the α derivative of the quadratic form, likewise the β derivative getting

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad = \quad \sum_i \; C_i' \begin{bmatrix} g_i \\ s_i \end{bmatrix} \; + \; C_i'' \left\{ \begin{bmatrix} \frac{\partial}{\partial \alpha} \\ \frac{\partial}{\partial \beta} \end{bmatrix} (\alpha g_i \; + \; \beta s_i) \right\} (\alpha g_i \; + \; \beta s_i) \tag{23}$$

resulting in a 2×2 set of equations to solve for α and β.

$$\left\{ \sum_i C_i'' \left[\begin{pmatrix} g_i \\ s_i \end{pmatrix} \begin{pmatrix} g_i & s_i \end{pmatrix} \right] \right\} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad = \quad - \sum_i C_i' \begin{bmatrix} g_i \\ s_i \end{bmatrix} \tag{24}$$

The solution of any 2×2 set of simultaneous equations is generally trivial. The only difficulties arise when the determinant vanishes which here is easy (luckily) to

understand. Generally the gradient should not point in the direction of the previous step if the previous move went the proper distance. Hence the determinant should not vanish. Practice shows that the determinant will vanish when all the inputs are zero, and it may vanish if you do so many iterations that you should have stopped already, in other words when the gradient and previous step are both tending to zero.

Using the newly found (α, β), update the residual $\bar{r}_i$ at each location (and update the model). Then go back to re-evaluate C_i' and C_i'' at the new r_i locations. Iterate.

In what way do we hope/expect this new bivariate solver embedded in a conjugate direction solver to perform better than old IRLS solvers? After paying the inevitable price, a substantial price, of computing $\mathbf{F}'\mathbf{r}$ and $\mathbf{F}\ \Delta\mathbf{m}$ the iteration above does some serious thinking, not a simple linearization, before paying the price again.

If the convex function $C(r)$ were least squares, subsequent iterations would do nothing. Although the Taylor series of the second iteration would expand about different residuals r_i than the first iteration, the new second order Taylor series are the exact representation of the least squares penalty function, i.e. the same as the first – so the next iteration goes nowhere.

Will this computational method work (converge fast enough) in the $L1$ limit? I don't know. Perhaps we'll do better to approach that limit (if we actually want that limit) via gradually decreasing the threshold R.

BLOCKY LOGS: BOTH FITTING AND REGULARIZATION

Here we set out to find blocky functions, such as well logs. We will do data fitting with a somewhat $L2$-like convex penalty function while doing model styling with a more $L1$-like function. We might define the composite norm threshold residual R_d for the data fitting at the 60th percentile, and that for regularization seeking spiky models (with blocky integrals) as R_m at the 5th percentile.

The data fitting goal and the model regularization goal at each z is independent from that at all other z values. The fitting goal says the reflectivity $m(z)$ should be equal to its measurement $d(z)$ (the seismogram). The model styling goal says the reflectivity $m(z)$ should vanish.

$$0 \approx r_d(z) = m(z) - d(z) \tag{25}$$

$$0 \approx r_m(z) = \epsilon\, m(z) \tag{26}$$

These two goals are in direct contradiction to each other. With the L2 norm the answer would be simply $m = d/(1+\epsilon^2)$. With the L1 norm, the answer would be either $m = d$ or $m = 0$ depending on the numerical choice of ϵ. Let us denote the convex function and its derivatives for data space at the residual as (B, B', B'') and for model space as (C, C', C''). Remember, m and d, while normally vectors, are here

scalars (independently for each z).

loop over all time points {
  $m = d/(1+\epsilon^2)$ # These are scalars!
  loop over non-linear iterations {
    $r_d = m - d$
    $r_m = m$
    Get derivatives of hybrid norm $B'(r_d)$ and $B''(r_d)$ for data goal.
    Get derivatives of hybrid norm $C'(r_m)$ and $C''(r_m)$ for model goal.
    # Plan to find α to update $m = m + \alpha$
    # Taylor series for data penalty $N(r_d) = B + B'\alpha + B''\alpha^2/2$
    # Taylor series for model penalty $N(r_m) = C + C'\alpha + C''\alpha^2/2$
    # $0 = \frac{\partial}{\partial \alpha}(N(r_d) + \epsilon N(r_m))$
    $\alpha = -(B' + \epsilon C')/(B'' + \epsilon C'')$
    $m = m + \alpha$
    } end of loop over non-linear iterations
  } end of loop over all time points

To help us understand the choice of parameters R_d, R_m, and ϵ, We examine the theoretical relation between m and d implied by the above code as a function of ϵ and R_m at $R_d \to \infty$, in other words, when the data has normal behavior and we are mostly interested in the role of the regularization drawing weak signals down towards zero. The data fitting penalty is $B = (m-d)^2/2$ and its derivative $B' = m - d$. The derivative of the model penalty (from equation (15)) is $C' = m/\sqrt{1 + m^2/R_m^2}$. Setting the sum of the derivatives to zero we have

$$0 \quad = \quad B' + \epsilon C' \quad = \quad m \; - \; d \; + \frac{\epsilon m}{\sqrt{1 + m^2/R_m^2}} \tag{27}$$

This says m is mostly a little smaller than d, but it gets more interesting near $(m, d) \approx 0$. There the slope $m/d = 1/(1+\epsilon)$ which says an $\epsilon = 4$ will damp the signal (where small) by a factor of 5. Moving away from $m = 0$ we see the damping power of ϵ diminishes uniformly as m exceeds R_m.

UNKNOWN SHOT WAVEFORM

A one-dimensional seismogram $d(t)$ is unknown reflectivity $c(t)$ convolved with unknown source waveform $s(t)$. The number of data points ND≈NC is less than the number of unknowns NC+NS. Clearly we need a "smart" regularization. Let us see how this problem can be set up so reflectivity $c(t)$ comes out with sparse spikes so the integral of $c(t)$ is blocky.

This is a nonlinear problem because the convolution of the unknowns is made of their product. Nonlinear problems elicit well-warranted fear of multiple solutions leading to us getting stuck in the wrong one. The key to avoiding this pitfall is

starting "close enough" to the correct solution. The way to get close enough (besides luck and a good starting guess) is to define a linear problem that takes us to the neighborhood where a nonlinear solver can be trusted. We will do that first.

Block cyclic solver

In the Block Cyclic solver (I hope this is the correct term.), we have two half cycles. In the first half we take one of the variables known and the other unknown. We solve for the unknown. In the next half we switch the known for the unknown. The beauty of this approach is that each half cycle is a linear problem so its solution is independent of the starting location. Hooray! Even better, repeating the cycles enough times should converge to the correct solution. Hooray again! The convergence may be slow, however, so at some stage (maybe just one or two cycles) you can safely switch over to the nonlinear method which converges faster because it deals directly with the interactions of the two variables.

We could begin from the assumption that the shot waveform is an impulse and the reflectivity is the data. Then either half cycle can be the starting point. Suppose we assume we know the reflectivity, say $\mathbf{c}$, and solve for the shot waveform $\mathbf{s}$. We use the reflectivity $\mathbf{c}$ to make a convolution matrix $\mathbf{C}$. The regression pair for finding $\mathbf{s}$ is

$$\mathbf{0} \quad \approx \quad \mathbf{Cs} \;-\; \mathbf{d} \tag{28}$$

$$\mathbf{0} \quad \approx \quad \epsilon_s \mathbf{I}\, \mathbf{s} \tag{29}$$

These would be solved for $\mathbf{s}$ by familiar least squares methods. It's a very easy problem because $\mathbf{s}$ has many fewer components than $\mathbf{c}$. Now with our source estimate $\mathbf{s}$ we can define the operator $\mathbf{S}$ that convolves it on reflectivity $\mathbf{c}$.

The second half of the cycle is to solve for the reflectivity $\mathbf{c}$. This is a little trickier. The data fitting may still be done by an $L2$ type method, but we need something like an $L1$ method for the regularization to pull the small values closer to zero to yield a more spiky $c(t)$.

$$\mathbf{0} \quad \approx_{L2} \quad \mathbf{r}_d \quad = \quad \mathbf{Sc} \;-\; \mathbf{d} \tag{30}$$

$$\mathbf{0} \quad \approx_{L1} \quad \mathbf{r}_m \quad = \quad \mathbf{I}\, \mathbf{c} \tag{31}$$

Normally we expect an ϵ_c in equation (31) but now it comes in later. (It might seem that the regularization (29) is not necessary, but without it, $\mathbf{c}$ might get smaller and smaller while $\mathbf{s}$ gets larger and larger. We should be able to neglect regression (29) if we simply rescale appropriately at each iteration.) We can take the usual L2 norm to define a gradient vector for model perturbation $\Delta\mathbf{c} = \mathbf{S}'\mathbf{r}_d$. From it we get the residual perturbation $\Delta\mathbf{r}_d = \mathbf{S}\Delta\mathbf{c}$. We need to find an unknown distance α to move in those directions. We take the norm of the data fitting residual, add to it a bit ϵ of the model styling residual, and set the derivative to zero.

$$\mathbf{0} \quad = \quad \frac{\partial}{\partial \alpha} \left[N_d(\mathbf{r}_d + \alpha\, \Delta\mathbf{r}) \;+\; \epsilon N_m(\mathbf{c} + \alpha \Delta\mathbf{c}) \right] \tag{32}$$

We need derivatives of each norm at each residual. We base these on the convex function $C(r)$ of the Hybrid norm. Let us call these A_i for the data fitting, and B_i for the model styling.

$$A_i = C(R_d, r_i) \tag{33}$$
$$B_i = C(R_m, c_i) \tag{34}$$

(Actually, we don't need A_i (because for Least Squares, $A_i' = r_i$ and $A_i'' = 1$), but I include it here in case we wish to deal with noise bursts in the data.) As earlier, expanding the norms in Taylor series, equation (32) becomes

$$0 \quad = \quad \sum_i A_i' \, \Delta r_i \; + \; \alpha \sum_i A_i'' \, \Delta r_i^2 \; + \; \epsilon \left(\sum_i B_i' \, \Delta c_i \; + \; \alpha \sum_i B_i'' \, \Delta c_i^2 \right) \tag{35}$$

which gives the α we need to update the model $\mathbf{c}$ and the residual $\mathbf{r}_d$.

$$\alpha \quad = \quad - \frac{\sum_i A_i' \, \Delta r_i \; + \; \epsilon \sum_i B_i' \, \Delta c_i}{\sum_i A_i'' \, \Delta r_i^2 \; + \; \epsilon \sum_i B_i'' \, \Delta c_i^2} \tag{36}$$

This is the steepest descent method. For the conjugate directions method there is a 2×2 equation like equation (24).

Non-linear solver

The non-linear approach is a little more complicated but it explicitly deals with the interaction between $\mathbf{s}$ and $\mathbf{c}$ so it converges faster. We represent everything as a "known" part plus a perturbation part which we will find and add into the known part. This is most easily expressed in the Fourier domain.

$$0 \quad \approx \quad (S + \Delta S)(C + \Delta C) \; - \; D \tag{37}$$

Linearize by dropping $\Delta S \Delta C$.

$$0 \quad \approx \quad S \; \Delta C \; + \; C \; \Delta S \; + \; (CS - D) \tag{38}$$

Let us change to the time domain with a matrix notation. Put the unknowns ΔC and ΔS in vectors $\tilde{\mathbf{c}}$ and $\tilde{\mathbf{s}}$. Put the knowns C and S in convolution matrices $\mathbf{C}$ and $\mathbf{S}$. Express $CS - D$ as a column vector $\bar{\mathbf{d}}$. Its time domain coefficients are $d_0 = c_0 s_0 - d_0$ and $d_1 = c_0 s_1 + c_1 s_0 - d_1$, etc. The data fitting regression is now

$$\mathbf{0} \quad \approx \quad \mathbf{S}\tilde{\mathbf{c}} + \mathbf{C}\tilde{\mathbf{s}} + \bar{\mathbf{d}} \tag{39}$$

This regression is expressed more explicitly below.

$$\mathbf{0} \quad \approx \quad \begin{bmatrix} s_0 & . & . & . & . & . & . & c_0 & . & . \\ s_1 & s_0 & . & . & . & . & . & c_1 & c_0 & . \\ s_2 & s_1 & s_0 & . & . & . & . & c_2 & c_1 & c_0 \\ . & s_2 & s_1 & s_0 & . & . & . & c_3 & c_2 & c_1 \\ . & . & s_2 & s_1 & s_0 & . & . & c_4 & c_3 & c_2 \\ . & . & . & s_2 & s_1 & s_0 & . & c_5 & c_4 & c_3 \\ . & . & . & . & s_2 & s_1 & s_0 & c_6 & c_5 & c_4 \\ . & . & . & . & . & s_2 & s_1 & . & c_6 & c_5 \\ . & . & . & . & . & . & s_2 & . & . & c_6 \end{bmatrix} \begin{bmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \\ \tilde{c}_3 \\ \tilde{c}_4 \\ \tilde{c}_5 \\ \tilde{c}_6 \\ \hline \tilde{s}_0 \\ \tilde{s}_1 \\ \tilde{s}_2 \end{bmatrix} + \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \\ d_8 \end{bmatrix} \tag{40}$$

The model styling regression is simply $0 \approx C + \Delta C$, which in familiar matrix form is

$$\mathbf{0} \quad \approx \quad \mathbf{I}\,\tilde{\mathbf{c}} + \bar{\mathbf{c}} \tag{41}$$

It is this regression, along with a composite norm and its associated threshold that makes $c(t)$ come out sparse. Now we have the danger that $\mathbf{c} \to \mathbf{0}$ while $\mathbf{s} \to \infty$ so we need one more regression

$$\mathbf{0} \quad \approx \quad \mathbf{I}\,\tilde{\mathbf{s}} + \bar{\mathbf{s}} \tag{42}$$

We can use ordinary least squares on the data fitting regression and the shot waveform regression. Thus

$$\mathbf{0} \quad \approx \quad \begin{bmatrix} \mathbf{r_d} \\ \mathbf{r_s} \end{bmatrix} \;=\; \begin{bmatrix} \bar{\mathbf{S}} & \bar{\mathbf{C}} \\ \mathbf{0} & \bar{\mathbf{I}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{c}} \\ \tilde{\mathbf{s}} \end{bmatrix} \;+\; \begin{bmatrix} \bar{\mathbf{d}} \\ \bar{\mathbf{s}} \end{bmatrix} \tag{43}$$

$$\mathbf{0} \quad \approx \quad \mathbf{r} \;=\; \mathbf{F}\mathbf{m} \;+\; \mathbf{d} \tag{44}$$

The model styling regression is where we seek spiky behavior.

$$\mathbf{0} \quad \approx \quad \mathbf{r_c} \;=\; \mathbf{I}\,\tilde{\mathbf{c}} \;+\; \bar{\mathbf{c}} \tag{45}$$

The big picture is that we minimize the sum

$$\min_{\tilde{\mathbf{c}},\tilde{\mathbf{s}}} \quad N_d(\mathbf{r}) \;+\; \epsilon \mathbf{N_m}(\mathbf{r_c}) \tag{46}$$

Inside the big picture we have updating steps

$$\Delta\mathbf{m} \;=\; \mathbf{F}'\mathbf{r} \tag{47}$$

$$\mathbf{g_d} = \Delta\mathbf{r} \;=\; \mathbf{F}\Delta\mathbf{m} \tag{48}$$

We also have a gradient for changing $\tilde{\mathbf{c}}$, namely $\mathbf{g_c} = -\bar{\mathbf{c}}$. (I need to be sure $\mathbf{g_d}$ and $\mathbf{g_c}$ are commensurate. Maybe need an ϵ here.) One update step is to choose a line search for α

$$\min_{\alpha} \quad N_d(\mathbf{r} + \alpha\mathbf{g_d}) \;+\; \epsilon \mathbf{N_m}(\bar{\mathbf{c}} + \alpha\mathbf{g_c}) \tag{49}$$

That was steepest descent. The extension to conjugate direction is straightforward.

As with all nonlinear problems there is the danger of bizarre behavior and multiple minima. To avoid frustration, while learning you should spend about half of your effort directed toward finding a good starting solution. This normally amounts to defining and solving one or two linear problems. In this application we might get our starting solution for $s(t)$ and $c(t)$ from conventional deconvolution analysis, or we might get it from the block cyclic solver.