Reconstructing measurements on a fine grid using regularization

M. Sadgrove
Center for Photonic Innovations, The university of Electro-communications, 1-5-1 Chofugaoka, Chofu, Japan[*]

(Dated: April 3, 2013)

I describe a method for reconstructing a set of measurements on a finer grid than that on which the measurements were taken using Tikhonov regularization. This method is useful when a smooth curve through a set of data points is required, but no analytical model exists.

I. INTRODUCTION

Often when processing data, we wish to draw a smooth curve through our measured points, either to illustrate the trend, or to allow easier interpolation between data points for someone viewing the data. If one has a theoretical model for the data which can be fitted to the measurements, this is the best way to provide such a smooth curve. However, in the case where no theory exists or when numerical solutions are too time intensive to allow fitting, other options must be pursued. Fitting of high order polynomials or spline curves can provide a fair match to the data in some cases. However, all too often such fitting produces curves with features not found in the data itself. Using such curves for interpolation between data points is thus fraught and open to accusations of being unscientific.

The purpose of this note is to make the reader aware of another option which lies conceptually between simple interpolation on one hand and a fitted curve on the other. In particular, I will describe how to “fill in” the points between the data on a mesh arbitrarily finer than the grid on which the data was taken. The method of filling in the points is to essentially perform an (inverse) linear transformation on the data which maps it onto the denser grid along with a regularization procedure which enforces smoothness of the solution by minimizing the value of arbitrary order derivatives of the solution.

The procedure is very close, but not identical, to a least squares fit with regularization. The difference is that the linear transform in question is not one mapping the data onto a straight line or curve, but rather a decimation transform which thins out the data on some hypothetical denser grid to give the measurements actually obtained (this will be made formal below; also see Fig. 1). This method was first described to me by Prof. Sze Tan of the Physics Department at Auckland University, and left to me as an excercise to implement. Although it is a fairly natural extension of regularized least squares fitting, I have not seen it described in any textbooks or papers. For this reason I describe the technique in detail below. Readers well versed in the linear algebra formulation of least squares can probably reproduce the technique from my simple description above. Additionally, the algorithm may have something in common with so called gridding or re-gridding procedures, which take data sampled on a non-uniform grid and reconstruct it on a uniform one. For the reader not familiar with either of these methods, this note may serve as an introduction to regularization techniques via a simple example. Beware, however, that it assumes familiarity with the linear algebra formulation of inverse problems. In principle, the first three chapters of the textbook for the University of Auckland “Inverse problems” course [1] should be referred to. The complete treatment of this very interesting area of applied mathematics is in no way reproduced here.

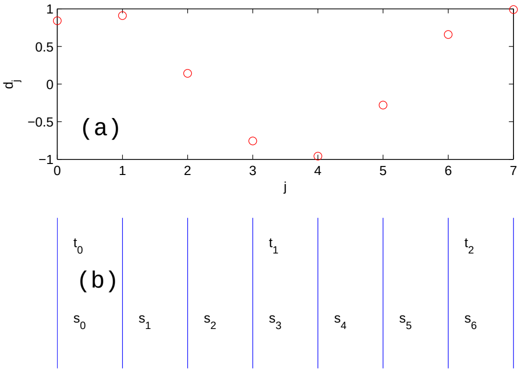

FIG. 1. Simple illustration of the fine-grid on which the reconstruction is to be performed.

II. STATEMENT OF THE PROBLEM

We have a collection of data $\{d_0, d_1, \dots, d_m\} \equiv \mathbf{d}$ which was measured at regularly spaced points $\{t_0, t_1, \dots, t_m\} \equiv \mathbf{t}$ on some grid, be it spatial, temporal or in the frequency domain, etc. Note that we will assume all lists such as $\mathbf{d}$ and $\mathbf{t}$ to be column vectors in what follows. Additionally, note that the requirement of equally spaced points can be relaxed in principle, but we assume equal spacing in this note so that the treatment is simple. We would like to reconstruct the data on a denser grid $\mathbf{s} \equiv \{s_0, s_1, \dots, s_n\}$, where $t_j = s_{Nj}$ for some integer N. Then we have $n = Nm$ and N may be seen to be the increase in density of the grid $\mathbf{s}$ in comparison to $\mathbf{t}$. The above concepts are depicted in Fig. 1, and in particular, the relationship between grids $\mathbf{t}$ and $\mathbf{s}$ are shown for $n = 6$ in Fig. 1(b).

Let us now define the reconstructed data $\mathbf{r} \equiv \{r_0, r_1, \dots, r_n\}$ which is sampled N times more densely than **d**. Although we do not know the contents of **r**, we know that it is related to **d** by the equation

$$r_{Nj} = d_j, \forall\, 0 \leq j \leq m. \qquad (1)$$

Our problem is the following

Statement of problem 1 *Given Eq. 1 and some minimal assumptions about the nature of* **r***, how can we solve for* **r***?*

Before we proceed, we will define our assumptions regarding **r**. Because we wish to use the method under discussion to produce a smooth curve through our data, we will require the following of **r**:

Assumption 1 *We require that* **r** *is smooth, that is, its first, second, …nth derivatives ($n = 2$ is sufficient for most practical purposes) are as small as possible while still being in good agreement with the data* **d***.*

Although the definition above may seem vague, in practice it is good enough to produce a regularized result for **r**. We will discuss the "as small as possible …" caveat below.

III. A SOLUTION OF THE PROBLEM: TIKHONOV REGULARIZED INVERSE MAPPING

A. Linear algebra representation of the problem

We now formally define the problem using the language of linear algebra. First we note the Eq. 1 can be represented as a matrix equation as follows: Let $\hat{M}$ be an $m \times n$ matrix which contains a single 1 in each row, and zeros everywhere else. More specifically, if the components of $\hat{M}$, $M_{i,j}$, are defined as follows

$$M_{i,j} = \delta_{Ni,j}, \qquad (2)$$

then it may be seen that Eq. 1 may be written in matrix form as

$$\hat{M}\mathbf{r} = \mathbf{d}. \qquad (3)$$

The abstract description above may be made clearer by looking at a specfic example: the case where $m = 3$ and $n = 6$ (i.e. $N = 3$) as illustrated in Fig. 1. In this case, the explicit form of matrix equation 3 is given by

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \end{pmatrix}. \qquad (4)$$

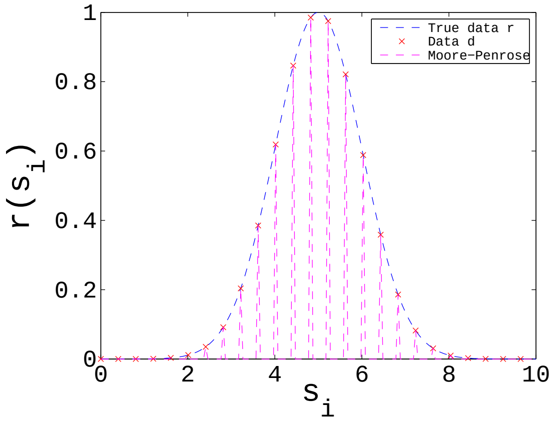

FIG. 2. The solution from the standard Moore-Penrose matrix inverse. Note that the solution is extremely spikey, but nonetheless good in a least squares sense.

B. Naïve solution: Moore-Penrose inverse (i.e. least squares solution)

Now that we have defined the problem explicitly as a matrix equation, it is tempting to see what happens if we simply solve Eq. 3 using the least squares method. This may be done by applying the Moore-Penrose pseudoinverse [2] $\hat{M}^+$ of $\hat{M}$ which is given by

$$\hat{M}^+ = (\hat{M}^t\hat{M})^{-1}\hat{M}^t, \qquad (5)$$

where $\hat{M}^t$ denotes the transpose of the matrix $\hat{M}$. We will use a slightly more complicated example than that given so far. Unlike in real life, in the example we actually start with knowledge of the "true data" **r**. For our example, this true data **r** is a Gaussian curve sampled on a grid of 200 points, as shown in Fig. 2 . We take $N = 8$, so the data **d** is sampled on a grid **t** 8 times as coarse as that of **r** (shown by crosses in Fig. 2).

Applying the Moore-Penrose pseudoinverse to Eq. 3 gives a solution $\mathbf{r}_{\mathrm{MP}}$

$$\mathbf{r}_{\mathrm{MP}} = (\hat{M}^t\hat{M})^{-1}\hat{M}^t\mathbf{d}. \qquad (6)$$

For the example, the solution $\mathbf{r}_{\mathrm{MP}}$ is shown as a magenta line in Fig. 2. It may be seen that this solution is not very useful, since values outside those defined by **d** are zero. Nonetheless, given no extra information about the solution **r**, this is indeed the most reasonable solution in a least-squares sense. However, we have assumed that the solution is smooth. How can we use this information to make a better solution?

C. Regularized solution

We now show how to apply a Tikhonov regularized solution which penalizes non smooth **r**. First we need to decide how to characterize the smoothness of $\mathbf{r}$. Because $\mathbf{r}$ is discrete, we need a numerical approximation to derivation. This is provided by matrices of the following form. First, the first derivative of $\mathbf{r}$ is approximated by applying the $(n-1)\times n$ matrix

$$\hat{L}_1 = \frac{1}{\Delta s}\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & -1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & -1 & 0 & 0 & \dots \\ & & & & & & \ddots \end{pmatrix}, \qquad (7)$$

where Δs is the spacing between points on the grid $\mathbf{s}$. As we will discuss below, we will concern ourselves only with the norm of the matrix $\hat{L}$ after it has been applied to $\mathbf{r}$. For this reason, although the matrix is $(n-1)\times n$, it is possible to add an nth row filled with zeros to make the matirx $n\times n$ without changing its characteristics in terms of our solution.

The second derivative can be discretely approximated by applying the $(n-2)\times n$ matrix

$$\hat{L}_1 = \frac{1}{(\Delta s)^2}\begin{pmatrix} 1 & -2 & 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & -2 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & -2 & 1 & 0 & \dots \\ & & & & & & \ddots \end{pmatrix}. \qquad (8)$$

As discussed above, we could pad the matrix with two final rows of zeros to make its dimensions $n\times n$ without changing its behaviour in terms of our solution.

We will only use the first and second derivatives in our example below, but higher order generalizations are of course possible. For the following formal analysis, we will assume that the smoothing operator is a single $n\times n$ matrix $\hat{L}$ representing a derivative of arbitrary order. The assumption of an $n\times n$ matrix is for simpicity of notation and is justified by the discussion of zero-padding above.

To proceed, we note that the norm of the vector $\hat{L}\mathbf{r}$ tells us something about the smoothness of the solution $\mathbf{r}$. Specifically, if we can minimize the squared norm $||\hat{L}\mathbf{r}||^2$, we have found the smoothest solution for a certain $\hat{L}$. (Note that the notation $||\mathbf{v}||$ for some vector $\mathbf{v}$ indicates the usual Cartesian norm of $\mathbf{v}$ i.e. $||\mathbf{v}|| = \sqrt{(}\sum_j v_j^2))$. In general, we will consider a linear combination of $||\hat{L}_1\mathbf{r}||^2$ and $||\hat{L}_2\mathbf{r}||^2$. However, we also wish the solution to agree well with the data. From these requirements, it is reasonable that we wish to find a solution $\mathbf{r}$ which satisfies the following equation:

$$\mathbf{r}_\lambda = \text{arg.min.}\{\lambda^2||\hat{L}\mathbf{r}||^2 + ||\mathbf{d} - \hat{M}\mathbf{r}||^2\}, \qquad (9)$$

where λ is a constant which parameterizes the amount of smoothing of the solution, and arg. min. denotes that the solution is the argument $\mathbf{r}$ which gives a minimal solution for the given value of λ.

Assuming that the RHS of Eq. 9 has a minimum, elementary calculus tells us that it occurs when the derivatives with respect to each component r_j of $\mathbf{r}$ are zero. That is, to solve Eq. 9, we want to find $\mathbf{r}$ which satisfies the equation

$$\frac{\partial}{\partial r_j}\{\lambda^2||\hat{L}\mathbf{r}||^2 + ||\mathbf{r} - \hat{M}\mathbf{d}||^2\} = 0,\ \forall 0 < k < n. \qquad (10)$$

Inspecting the first term on the LHS of Eq. 10, we find that

$$\begin{aligned}\frac{\partial}{\partial r_j}\lambda^2||\hat{L}\mathbf{r}||^2 &= \lambda^2\frac{\partial}{\partial r_j}\sum_{k=0}^{n}\left(\sum_{\ell=0}^{n}L_{k,\ell}r_\ell\right)^2 \\ &= \lambda^2\sum_{k=0}^{n}\sum_{\ell,\ell'}^{n}\frac{\partial}{\partial r_j}L_{k,\ell}L_{k,\ell'}r_\ell r_{\ell'} \\ &= 2\lambda^2\sum_{k=0}^{n}L_{k,j}\left(\sum_{\ell=0}^{n}L_{k,\ell}r_\ell\right). \qquad (11)\end{aligned}$$

Note that the lower range for both ℓ and ℓ', omitted from the double sum for clarity, is 0. After some consideration, it may be seen that the final expression is the jth component of the vector $2\lambda^2\hat{L}^t\hat{L}\mathbf{r}$.

We now consider the second term on the RHS of Eq. 9. Using the fact that $||\mathbf{v}||^2 = \mathbf{v}^t\mathbf{v}$, we may write

$$\begin{aligned}||\mathbf{d} - \hat{M}\mathbf{r}||^2 &= (\mathbf{d} - \hat{M}\mathbf{r})^t(\mathbf{d} - \hat{M}\mathbf{r}) \\ &= \mathbf{d}^t\mathbf{d} - (\hat{M}\mathbf{r})^t\mathbf{d} \\ &\quad -\mathbf{d}^t(\hat{M}\mathbf{r}) + (\hat{M}\mathbf{r})^t(\hat{M}\mathbf{r}). \qquad (12)\end{aligned}$$

Then, considering the partial derivative $(\partial/\partial r_k)||\mathbf{d} - \hat{M}\mathbf{r}||^2$, the first term on the RHS of Eq. 12 has no r_k dependence and so goes to zero. The partial derivative of the final term on the last line of Eq. 12 has the form $(\partial/\partial r_k)||\hat{M}\mathbf{r}||^2$ and so by analogy with the result for $||\hat{L}\mathbf{r}||^2$ above, we find that it is $2\hat{M}^t\hat{M}\mathbf{r}$ in vector form. The remaining 2 middle terms in the RHS of Eq. 12 will now be considered. Specifically, we find for the second term:

$$\begin{aligned}\frac{\partial}{\partial r_j}(\hat{M}\mathbf{r})^t\mathbf{d} &= \frac{\partial}{\partial r_j}\left\{\sum_{k=0}^{n}\left(\sum_{\ell=0}^{n}M_{k,\ell}r_l\right)d_k\right\} \\ &= \left\{\sum_{k=0}^{n}\left(\sum_{\ell=0}^{n}\frac{\partial}{\partial r_j}M_{k,\ell}r_\ell\right)d_k\right\} \\ &= \sum_k M_{k,j}d_k. \qquad (13)\end{aligned}$$

The final term of Eq 13 may be seen to be the jth component of the vector $\hat{M}^t\mathbf{d}$. The partial derivative of the third term on the RHS of Eq. 12 is simply the transpose of the second term and must give the same result. Thus, putting all of the partial derivatives together in vector form, we can write Eq. 10 as

$$2\lambda^2\hat{L}^t\hat{L}\mathbf{r} - 2\hat{M}^t\mathbf{d} + 2\hat{M}^t\hat{M}\mathbf{r} = 0. \qquad (14)$$

Finally, rearranging gives

$$\mathbf{r}_\lambda = (\lambda^2\hat{L}^t\hat{L} + \hat{M}^t\hat{M})^{-1}\hat{M}^t\mathbf{d}. \qquad (15)$$

Note that if the regularization parameter is zero, we regain the Moore-Penrose inverse as expected.

The choice of regularization parameter λ is not completely trivial. For non-rigorous purposes, trial and error quickly produces reasonable fitting. To more carefully justify the choice of a particular value for lambda, the so-called "L-curve" which plots the smoothing norm $||\hat{L}\mathbf{r}||^2$ against the data misfit $||\mathbf{d} - \hat{M}\mathbf{r}||$ can be used (See [1], Chapter 3, Fig 3.2). I will not deal with such questions here, and only show the results of the regularization procedure in a few cases.

IV. EXAMPLE IMPLEMENTATION IN OCTAVE OR MATLAB

Since the solution given by Eq 15 requires inverting a matrix that scales as the square of the data set size, brief consideration suggests that the solution may become intractable for large data sets. However, all of the matrices used in the formulation are sparse, banded matrices. That is, most of the matrix entries are zero, except for a band around the leading diagonal. In this case, sparse matrix representations can be used by most modern linear algebra packages saving memory, and making it possible to perform the reconstruction discussed here even for data sets with thousands of points. We give an example program below which can run on either the free software package *Octave* or the commercial package *Matlab* and which uses sparse matrix representations to make the memory impact as small as possible.

```
% Show a simple example of Tikhonov regularization
% using sparse matrices

clear all
close all

s = linspace(0,10,200)';

% We use a simple Gaussian curve
% for demonstration purposes

% NB: We actually know the "true" data in this case
% but in the experiment there is no "true" data,
% only a reconstruction for given regularization
% parameter
r = exp(-(s-5).^2/2);

% Normalizing means the regularization parameter
% won't need to be too large
r = r/max(r);

% Plot the true data
figure(1)
clf
plot(s,r,'b--')

% Now sample the points on a coarser grid.
% This represents the data that we would
% actually have measured in an experiment
nn=8 % 8 times coarser than t
t = s(1:nn:end);
d = r(1:nn:end);

hold on
plot(t,d,'rx')
hold off

% First let's check what standard least squares
% gives by applying the Moore-Penrose inverse

% First, make the transfer matrix as a sparse matrix to save space/
% calculation time
m = length(d);
n = length(r);
M = sparse(1:m,1:nn:n,ones(length(m)),m,n);

% Now calculate the reconstruction,
% which should in principle be close to
% r. In practice, because we don't have enough
% information, we get zeros everywhere
% but at the spikes where t=s
```

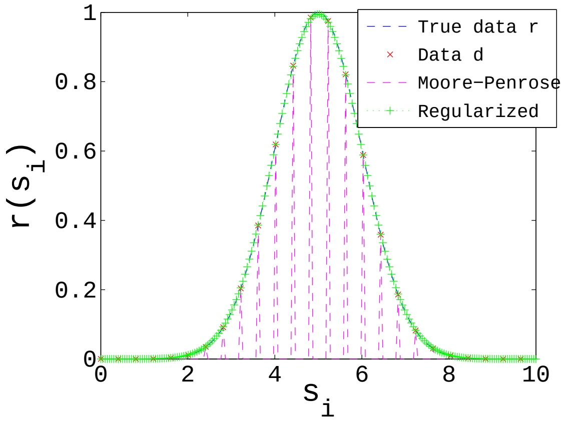

FIG. 3. Same as for Fig. 2, but with the regularized solution (with $\lambda_1 = 0.02$, $\lambda 2 = 0.005$) shown by green crosses. The solution matches the true data so well that it is difficult to see the true data, which is a blue line below the green crosses.

```
recon = M\d; % Least squares solution with least possible
     % non-zero entries

% Plotting the naive reconstruction
hold on
plot(s,recon,'m--')
hold off

xlabel('s_i')
ylabel('r(s_i)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Now let's apply Tikhonov regularization.
% First, we'll define the matrix to realize the
%Sobolev norm of order 1
% Note the use of sparse matrices here as for M above
L1 = (1/(s(2)-s(1)))*(-1*sparse(1:(n-1),1:(n-1),...
  ones(1,n-1),n-1,n) + sparse(1:(n-1),2:(n),...
  ones(1,n-1),n-1,n))
L2 = (1/(s(2)-s(1))^2)*(sparse(1:(n-2),1:(n-2),...
  ones(1,n-2),n-2,n) - 2* sparse(1:(n-2),...
  2:(n-1),ones(1,n-2),n-2,n)...
    + sparse(1:(n-2),3:(n),ones(1,n-2),n-2,n))

% Regularization constant
% We split the regularization constant into two parts,
% to separately deal with first and second order smoothing
lambda1 = 0.02
lambda2=0.005

% Finally, we perform the Tikhonov regularized reconstruction
recon_lambda = (lambda1^2*L1'*L1 + lambda2^2*L2'*L2 + M'*M)\(M'*d);

% And plot
hold on
plot(s,recon_lambda,'g+:')
hold off
legend('True data r','Data d'
```

The output of this program is shown in Fig 3. In the example, we use two regularization terms corresponding to the first and second derivative of $\mathbf{r}$ respectively. The values of the regularization constants were $\lambda_1 = 0.02$, $\lambda 2 = 0.005$.

Finally, in Fig. 4, I show data from an actual experiment with a regularized curve going through the data. This real world example shows the true value of regularized reconstruction methods. The relatively low frequency resolution of the optical multichannel analyser used to take the reflection data (red circles in Fig. 4) gives only a few points near the center of the peak. Compared with the transmission spectrum (measured on the Fourier transform spectrum analyser) shown in blue, the data resolution is not good. Without regularization, simply joining the circles would make a jagged peak. Using a regularized solution allows us to fit a smooth curve even without a physical model for the reflection peak. As can be seen, the regularization does not pass exactly through the points near the center of the peak - a feature which is due to the smoothing applied. This is the advantage of regularization over simple interpolation.

The Matlab code for reconstructing the OMA data is given in Appendix A

Appendix A: Matlab code for reconstruction of OMA data

```
% OMA data
dataBG = importdata('../TransmissionData20130227/ReflBackground_xm05_WorstPol_20130227.txt');
RBG = dataBG(:,2);
lambdar = dataBG(:,1);
DLr = (lambdar(2)-lambdar(1))
lb = find((lambdar)>700);
ub = find((lambdar)<900);
startind = lb(1)
endind = ub(end)

dataR = importdata('../TransmissionData20130227/ReflData_xm05_WorstPol_20130227.txt');
Rdata = dataR(:,2);

RNorm = Rdata(startind:endind) - RBG(startind:endind);
RNorm = (RNorm/max(RNorm)) * R765_mean

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The OMA data has a frequency resolution approx. 4 times worse than the
% FT. Here I will reconstruct the OMA data on a finer grid using a Tikhonov
% reconstruction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

lplot = lambdar(startind:endind)
lplot = lplot(:); % Need to have column vectors for standard linear algebra constructions

% Define resolution increase. Here we will choose 4 times the data resolution
nn = 4;

ldense = [];
for kk = 1:(length(lplot)-1)
    toadd = linspace(lplot(kk),lplot(kk+1),nn+1);
    ldense = [ldense toadd(1:nn)];
end
ldense = [ldense, lplot(end)];
ldense = ldense(:);

% Define the transfer matrix
m = length(lplot);
n = length(ldense);
whos
A = sparse(1:m,1:nn:n,ones(length(m)),m,n); % Sparse matrices save memory and time

% Define 1st and 2nd derivative matrices for the Sobolev norm
L1 = (1/(ldense(2)-ldense(1)))*(-1*sparse(1:(n-1),1:(n-1),ones(1,n-1),n-1,n) + ...
  sparse(1:(n-1),2:(n),ones(1,n-1),n-1,n))
L2 = (1/(ldense(2)-ldense(1))^2)*(sparse(1:(n-2),1:(n-2),ones(1,n-2),n-2,n) - ...
  2* sparse(1:(n-2),2:(n-1),ones(1,n-2),n-2,n)...
  + sparse(1:(n-2),3:(n),ones(1,n-2),n-2,n))

% Regularization constants: we define one for first derivative and one for
% second derivative
lambda1 = 0.01
lambda2=0.1

% Now the reconstruction. Note that the preferred solution is set to zero
% i.e. there is no preferred solution
Rrecon = inv(lambda1^2*L1'*L1 + lambda2^2*L2'*L2 + A'*A)*(A'*RNorm);
```

[*] mark@cpi.uec.ac.jp

[1] S.M. Tan, C. Fox, and G.K. Nicholls “Inverse problems”. May be downloaded from http://home.comcast.net/~szemengtan/ . Also see the website of Collin Fox’s Electronics 404 course at Otago University: http://elec.otago.ac.nz/w/index.php/ELEC_404

[2] See, for example, http://mathworld.wolfram.com/Moore-PenroseMatrixInverse

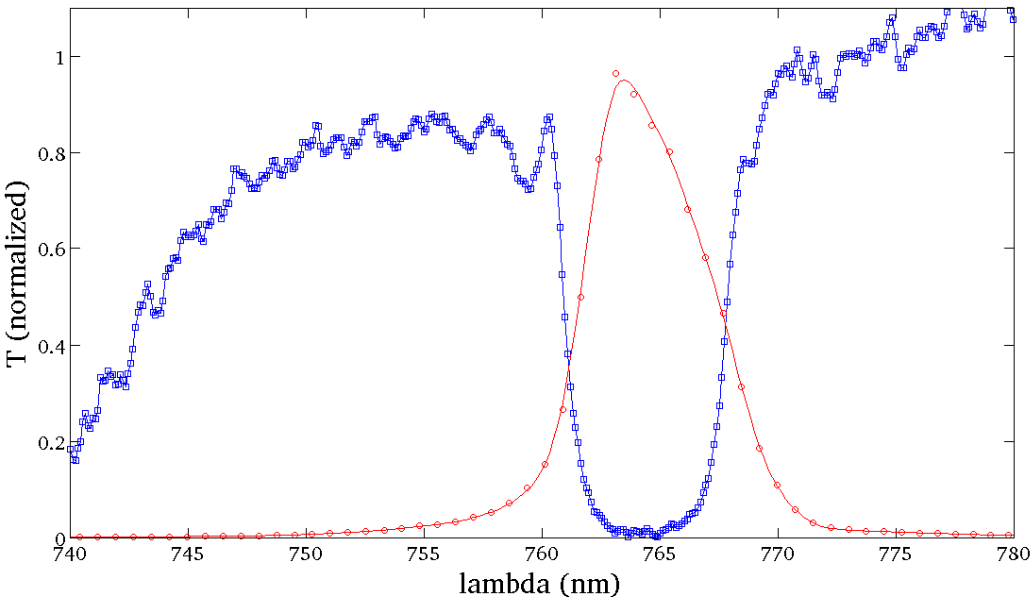

FIG. 4. Data from an experiment at the Center for Photonic Innovations. The red solid line is a regularized solution for the data shown by the red circles.