Course Name: Digital Signal Processing Design

Course Number and Section: **14:332:447:01**

Trend Extraction: Regularized and Sparse Whittaker-Henderson Methods

Submitted by: Lance Darragh

Introduction

We are going to investigate methods of data smoothing, generally used for trend extraction, known as the Whittaker-Henderson smoothing methods. They are commonly used today in financial markets, yet they can be applied to find data trends in any field where statistical data is given. Whittaker-Henderson smoothing methods were first developed for use in the actuarial sciences, and they are discrete-time versions of spline smoothing for equally spaced data. For more information on spline smoothing, see chapter 8 of [1]. Like all smoothing operations, they act as a lowpass filter, but they are actually designed using a highpass filter, along with a correction term that is used to minimize the spectral density of the output signal. We will describe the Whittaker-Henderson smoothing methods and observe their performance on a few sets of data.

Whittaker-Henderson Smoothing

We begin by defining the *performance index* as

$$\mathcal{J} = \sum_{n=0}^{N-1} w_n \,|y_n - x_n|^2 + \lambda \sum_{n=s}^{N-1} |\nabla^s x_n|^2 = \text{min} \tag{1}$$

where $\nabla^s x_n$ represents the backward-difference operator $\nabla x_n = x_n - x_{n-1}$ applied s times. The first term acts as an interpolation filter, and the second term is the correction term used to minimize the spectral energy density in the output. Most often, s equals 0, 1, 2, or 3, with s = 2 being the most common, and we'll be using s = 2 in the examples we do here. These can be though of as the first, second, and third derivatives of our data signal, and the s = 2 case can best be thought of as the acceleration rate of our data between samples. The corresponding difference filter is

$$D_s(z) = (1 - z^{-1})^s \tag{2}$$

and its impulse response is

$$d_s(k) = (-1)^k \binom{s}{k}, \quad 0 \le k \le s \tag{3}$$

Following an example from [1], we have for s = 1, 2, 3

$$\mathbf{d}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{d}_2 = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}, \quad \mathbf{d}_3 = \begin{bmatrix} 1 \\ -3 \\ 3 \\ -1 \end{bmatrix}$$

The second term in equation (1) can be described as

$$g_n = \nabla^s x_n = \sum_{k=\max(0,n-N+1)}^{\min(s,n)} d_s(k) x_{n-k}, \quad 0 \le n \le N-1+s \tag{4}$$

but, this differencing operation can actually be replaced by any other causal FIR highpass filter to obtain a similar result. Denoting this as d_n, and letting s denote the filter order, we have a more general version.

$$\mathcal{J} = \sum_{n=0}^{N-1} w_n |y_n - x_n|^2 + \lambda \sum_{n=s}^{N-1} |d_n * x_n|^2 = \min \tag{4}$$

Assuming N > s, we define the second term as

$$g_n = d_n * x_n = \sum_{k=\max(0,n-N+1)}^{\min(s,n)} d_k x_{n-k}, \quad 0 \le n \le N-1+s \tag{5}$$

and this can be expressed vectorially as $\mathbf{g} = D_s\mathbf{x}$. Here $\mathbf{x}$ is an N-dimensional input column vector, and $\mathbf{g}$ is an (N + s)-dimensional output vector. Ds is an (N + s) x N, convolution matrix. By the minimization term, we aim to obtain a sparse solution for $\mathbf{g} = D_s\mathbf{x}$, corresponding to a vector $\mathbf{g}$ with few nonzero values. The corresponding steady-state output vector can be expressed as a squared norm,

$$\sum_{n=s}^{N-1} [d_n * x_n]^2 = \sum_{n=s}^{N-1} g_n^2 = \mathbf{g}^T\mathbf{g} = \mathbf{x}^T(D^T D)\mathbf{x} \tag{6}$$

Using this we can rewrite our performance index as.

$$\mathcal{J} = (\mathbf{y} - \mathbf{x})^T W (\mathbf{y} - \mathbf{x}) + \lambda \mathbf{x}^T (D^T D)\mathbf{x} = \min \tag{10}$$

where W is a diagonal matrix of weighting coefficients. To solve this minimization condition, we take the derivate of J with respect to $\mathbf{x}$, and set this gradient equal to zero.

$$\frac{\partial \mathcal{J}}{\partial \mathbf{x}} = -2W(\mathbf{y} - \mathbf{x}) + 2\lambda (D^T D)\mathbf{x} = 0 \quad \Rightarrow \quad (W + \lambda D^T D)\mathbf{x} = W\mathbf{y} \tag{11}$$

Solving for $\mathbf{x}$, we get the solution

$$\hat{\mathbf{x}} = (W + \lambda D^T D)^{-1} W\mathbf{y} \tag{12}$$

In MATLAB this is solved efficiently using the backslash operator.

Regularization Filters

The Whittaker-Henderson smoothing methods, like spline smoothing methods, are examples of *regularization*. More generally, the performance index of equation (1) can be used for inverse filtering operations. Letting f_n and d_n represent FIR filters, we have

(13) $$\mathcal{J} = \sum_n |y_n - f_n * x_n|^2 + \lambda \sum_n |d_n * x_n|^2 = \min$$

This equation attempts to deconvolve the effect of f_n to solve for x_n. Using the convolution matrix forms, F and D, of f_n and d_n, we can rewrite this as

(14) $$\mathcal{J} = \|\mathbf{y} - F\mathbf{x}\|^2 + \lambda \|D\mathbf{x}\|^2 = (\mathbf{y} - F\mathbf{x})^T(\mathbf{y} - F\mathbf{x}) + \lambda \mathbf{x}^T(D^T D)\mathbf{x} = \min$$

Similar to how we took the gradient in equation (11), we have

(15) $$\frac{\partial \mathcal{J}}{\partial \mathbf{x}} = -2F^T(\mathbf{y} - F\mathbf{x}) + 2\lambda D^T D\mathbf{x} = 0\,,$$

with the result that,

(16) $$\hat{\mathbf{x}} = (F^T F + \lambda D^T D)^{-1} F^T \mathbf{y}$$

This method of regularization is knows as *Tikhonov regularization*, or *ridge regression*. However, the system of linear equations, $\mathbf{y} = F\mathbf{x}$ may not be explicitly solvable. In general, the system may be overdetermined, underdetermined, or rank defective. In fact, regularization was originally developed to handle problems that are ill-posed, inconsistent, overdetermined, or ill-conditioned. We will discuss the underdetermined and rank defective cases in the next project, but here we restrict our attention to the cases where the system is square and invertible or overdetermined but that F is full rank. For $\lambda = 0$, we have

(17) $$\hat{\mathbf{x}} = (F^T F)^{-1} F^T \mathbf{y}$$

In the square case,

(18) $$\hat{\mathbf{x}} = F^{-1}\mathbf{y}$$

In general, we model our signal as **y** = F**x** + **v**, and we want to estimate a solution for **x**, which gives us,

(19) $$\hat{\mathbf{x}} = F^{-1}\mathbf{y} = F^{-1}(F\mathbf{x} + \mathbf{v}) = \mathbf{x} + F^{-1}\mathbf{v}$$

In practice, a common problem is that F is ill-conditioned with a large condition number, which results in division by a very small number when solving for the inverse, magnifying the noise in the signal to such an extent that **x** is masked and rendered useless. This can also happen in the underdetermined case. To condition number of a matrix equation A**x** = **b** is defined as the maximum ration of the relative error in **x** to the relative error in **b**. Essentially, if there is a high condition number, when the data in **b** changes by a small amount, the resulting value of **x** will change more dramatically, resulting in a less-accurate, and possibly erroneous, solution. See [2] for more details on this topic. For our purposes, we note that the regularization term helps to provide a more well-conditioned inverse, providing a more accurate solution. For inverse filtering, we typically set D = I, the unit identity matrix, which gives the solution

(20) $$\hat{\mathbf{x}} = (F^T F + \lambda I)^{-1} F^T \mathbf{y}$$

To see how this helps to improve the condition number of the system, let λ_{max} and λ_{min} represent the maximum and minimum eigenvalues of F^TF. The condition number of F^TF would be $\lambda_{max}/\lambda_{min}$, and a highly ill-conditioned problem would have $\lambda_{max} >> \lambda_{min}$, resulting in erroneously large values in our solution. However, for $F^TF + \lambda I$ we have $(\lambda_{max} + \lambda)/(\lambda_{min} + \lambda)$ which, for an ill-conditioned problem, is much less than $\lambda_{max}/\lambda_{min}$, making our solution much more accurate. Although this is certainly a way of reducing the error in our solution, it also introduces some distortion and sometimes more smoothing than desired. Choosing the right value of λ usually involves some trial-and-error, and we'll explore this process in our examples.

Whittaker-Henderson Smoothing Summary

The sparsity of solutions for **g** = D_s**x** can be obtain in different ways depending on which value of s we choose. These correspond to the following optimization criteria. For a length-N signal of data points, y_n, with n from 0 to N -1,

(21) $$\mathcal{J} = \sum_{n=0}^{N-1} |y_n - x_n|^2 + \lambda \sum_{n=s}^{N-1} |\nabla^s x_n|^2 = ||\mathbf{y} - \mathbf{x}||_2^2 + \lambda ||D_s\mathbf{x}||_2^2 = \min \qquad (L_2)$$

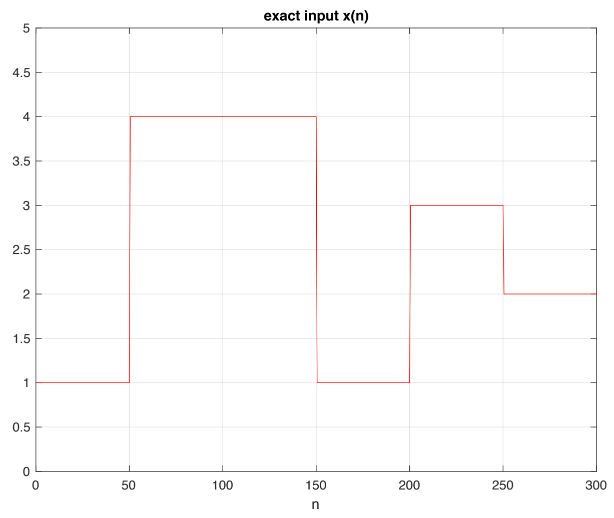
(22) $$\mathcal{J} = \sum_{n=0}^{N-1} |y_n - x_n|^2 + \lambda \sum_{n=s}^{N-1} |\nabla^s x_n|^1 = ||\mathbf{y} - \mathbf{x}||_2^2 + \lambda ||D_s\mathbf{x}||_1 = \min \qquad (L_1)$$

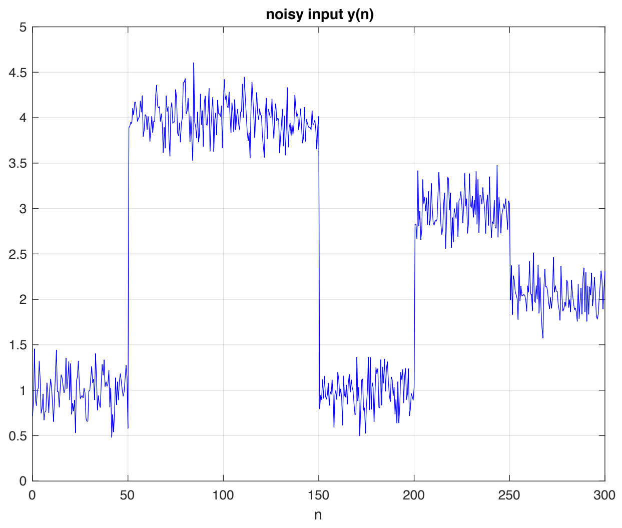
(23) $$\mathcal{J} = \sum_{n=0}^{N-1} |y_n - x_n|^2 + \lambda \sum_{n=s}^{N-1} |\nabla^s x_n|^0 = ||\mathbf{y} - \mathbf{x}||_2^2 + \lambda ||D_s \mathbf{x}||_0 = \min \qquad (L_0)$$

For the L_0 case, we can see that the solution will be minimized when it has the most values equal to zero. This is the sparsest solution. The L_1 case corresponds to having the highest sparsity when the solution is a straight line, and the L_2 case is sparsest when the solution has equal curvature. In general, these solutions have the highest sparsity for modeling signals composed of piecewise polynomials of degree s - 1. Higher values of s can be used, but they are not very common, and Whittaker and Henderson used s = 3 when they developed these methods.

Examples

First let us consider a flat-top signal and its observed noisy version, y_n, defined by $y_n = x_n + v_n$, where n goes from 0 to 600.





The solution to the L_2 problem is

(24) $$\mathbf{x} = (I + \lambda D_s^T D_s)^{-1} \mathbf{y}$$

which can be solved easily in MATLAB with the following command

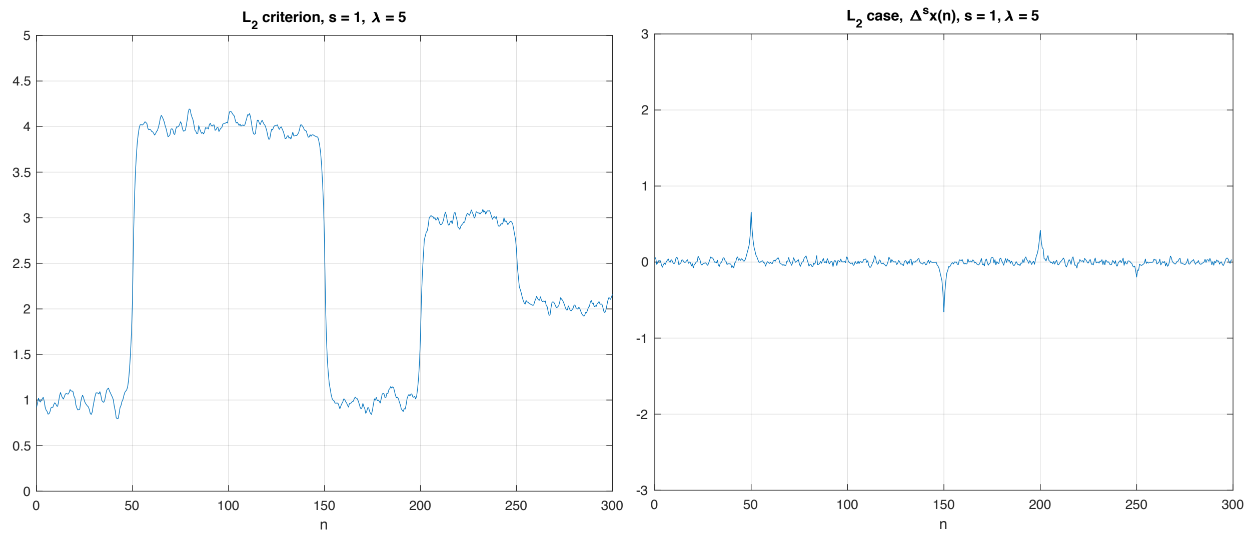
x = (speye(N) + λD$_s$'Ds)**y**;

where speye(N) creates the N x N, sparse identity matrix, which is the unit identity matrix containing only the nonzero values. The D_s matrix can be formed sparsely as

Ds = diff(speye(N), s);

Here N is the length of our input signal, 601.

We'll first use the L_2 criterion with s = 1, and $\lambda = 5$ to estimate the signal x_n. We'll also show the s-differenced signal $\nabla^s x_n = Ds*x_n$.



As we can see, the L_2 criterion worked reasonably well, and the first-difference signal is very small and of minimal energy. However, since its values are mostly non-zero it is not considered a sparse solution. Next we'll try the L_1 criterion with the same parameters. We'll solve this using two different methods. First we'll solve it with the CVX package, a package that extends the language of MATLAB. Then we'll solve it using the iteratively reweighted least squares (IRLS) method. The basic idea of the IRLS method is to replace the more general L_p norm with an L_2 norm because the L_2 norm can be solved iteratively. For any value $0 \leq p \leq 2$, not necessarily an integer, we let q = 2 - p and write (for $x \neq 0$),

$$(25) \qquad |x|^p = \frac{|x|^2}{|x|^q}$$

Now, to handle the case where x = 0, we define epsilon to be a sufficiently small number and write,

$$(26) \qquad |x|^p = \frac{|x|^2}{|x|^q} \approx \frac{|x|^2}{|x|^q + \varepsilon}$$

For the L$_p$ norm of a vector $\mathbf{x} \in \mathbb{R}^N$,

$$(27) \qquad \|\mathbf{x}\|_p^p = \sum_{i=0}^{N-1} |x_i|^p \approx \sum_{i=0}^{N-1} \frac{|x_i|^2}{|x_i|^q + \varepsilon} = \mathbf{x}^T W(\mathbf{x})\mathbf{x}$$

with

$$(28) \qquad W(\mathbf{x}) = \operatorname{diag}\left[\frac{1}{|\mathbf{x}|^q + \varepsilon}\right] = \operatorname{diag}\left[\frac{1}{|x_0|^q + \varepsilon}, \frac{1}{|x_1|^q + \varepsilon}, \ldots, \frac{1}{|x_{N-1}|^q + \varepsilon}\right]$$

The L$_p$ regularized problems can be written as

$$(29) \qquad \mathcal{J} = ||\mathbf{y} - \mathbf{x}||_2^2 + \lambda||D_s\mathbf{x}||_p^p = ||\mathbf{y} - \mathbf{x}||_2^2 + \lambda\, \mathbf{x}^T D_s^T W(D_s\mathbf{x})\, D_s\mathbf{x} = \min$$

Which gives us the following iterative algorithm,

$$(30) \qquad \begin{aligned} &\text{for } k = 1, 2, \ldots, K, \text{ do:} \\ &\quad W_{k-1} = W(D_s\mathbf{x}^{(k-1)}) \\ &\quad \mathbf{x}^{(k)} = \arg\min_{\mathbf{x}} ||\mathbf{y} - \mathbf{x}||_2^2 + \lambda\, \mathbf{x}^T D_s^T W_{k-1} D_s\mathbf{x} \end{aligned}$$

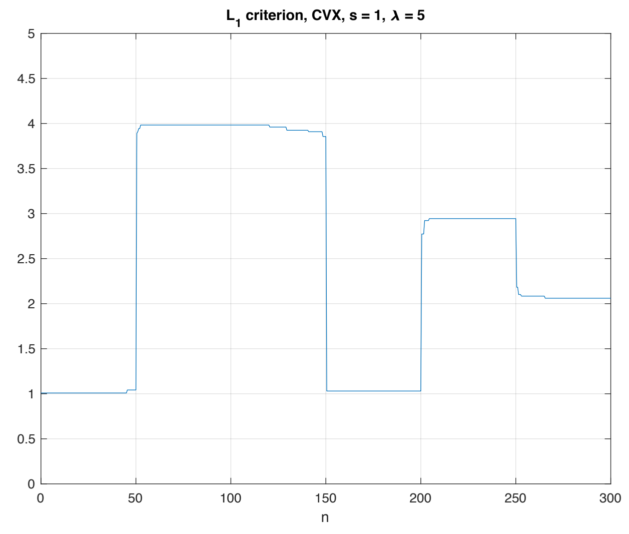
The solution at the kth step is

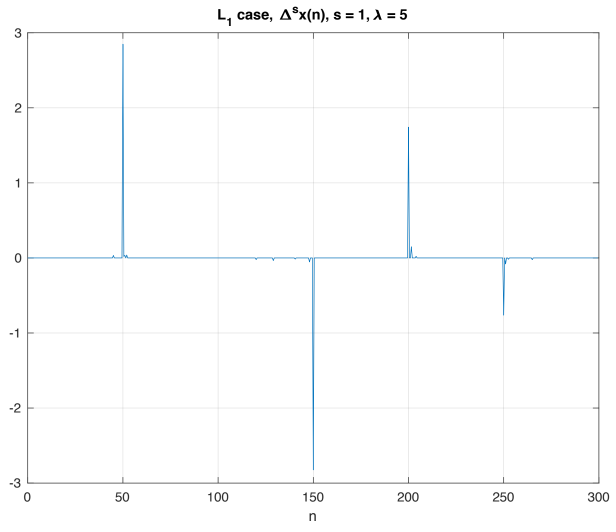
$$(31) \qquad \mathbf{x}^{(k)} = (I + \lambda D_s^T W_{k-1} D_s)^{-1}\mathbf{y}$$

and we initialize **x** with the ordinary least-squares solution,

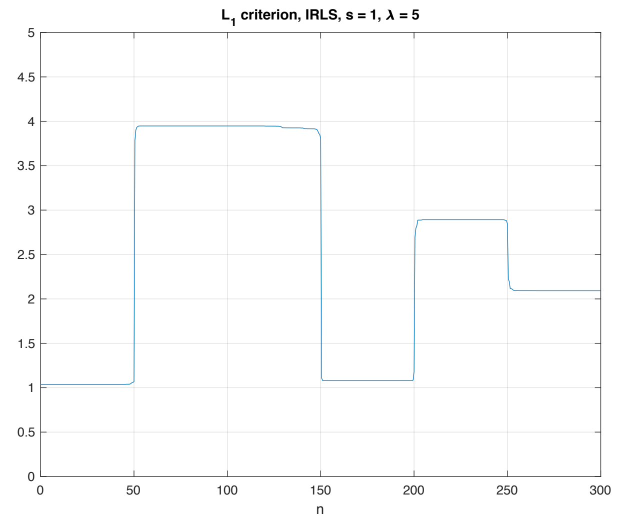
$$(32) \qquad \mathbf{x}^{(0)} = (I + \lambda D_s^T D_s)^{-1}\mathbf{y}$$

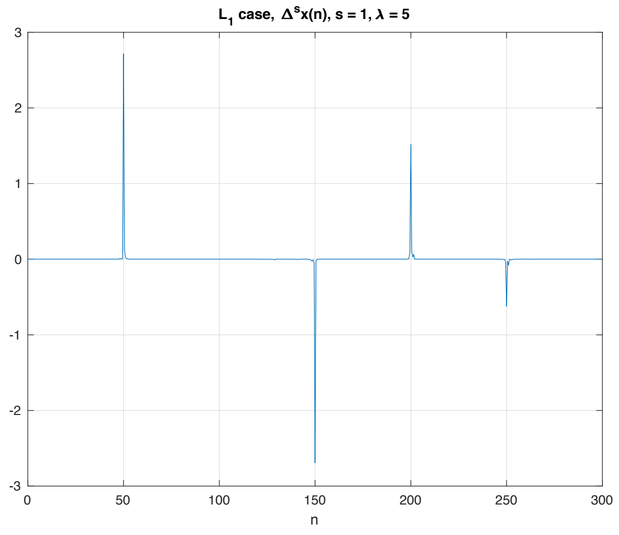
The CVX package can be found at, http://cvxr.com/cvx/, and the code to implement their algorithm, along with the others in this example can be found in the Appendix. Using this package we obtain the following results,



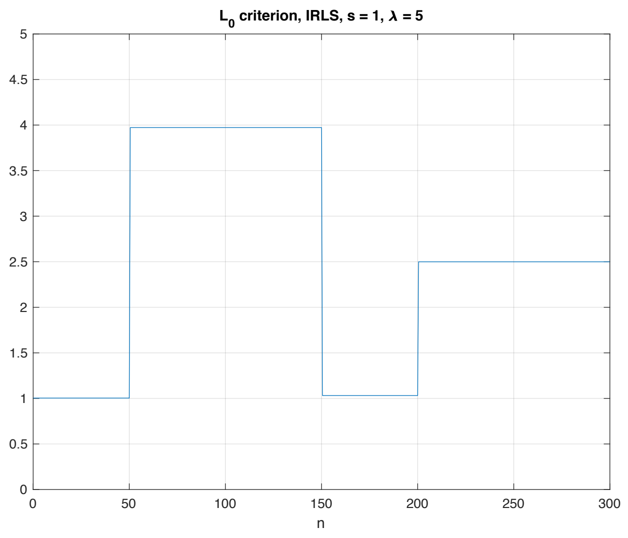


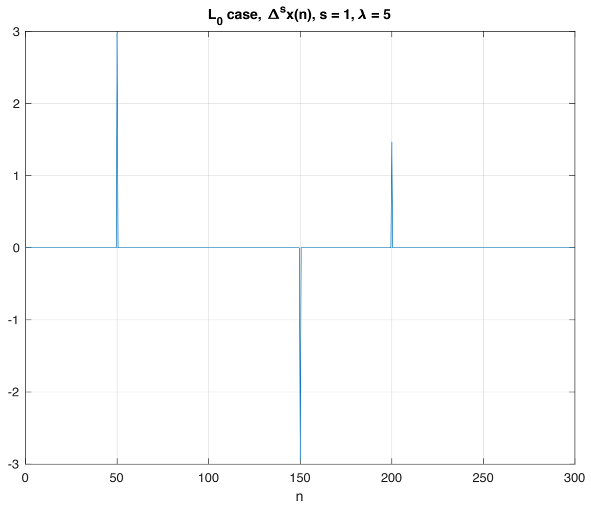
which we can see are very accurate and sparse by the minimization condition. With the IRLS method,



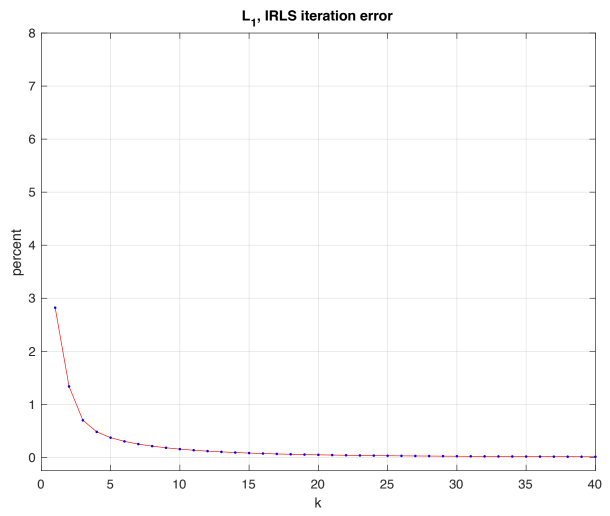


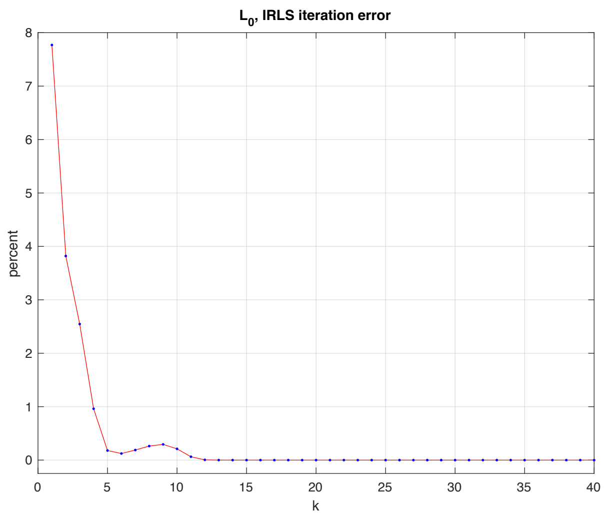
we obtain a very similar result. We will also use the IRLS algorithm to solve the L_0 problem,





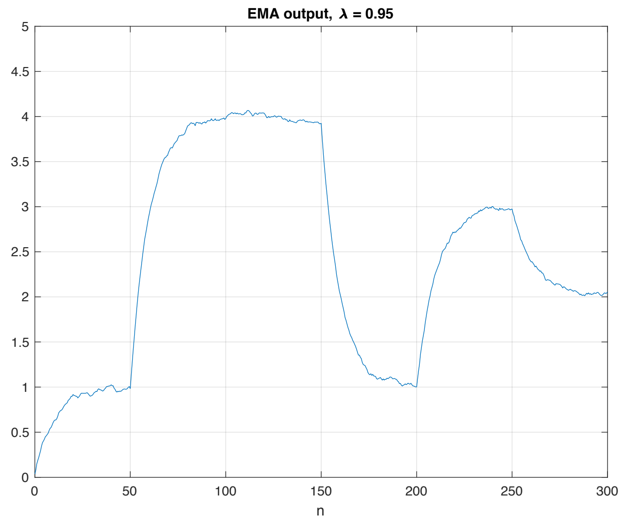
which gives us an even better result. The respective abilities for these different criterion to reduce noise while still accurately extracting the noise-free signal are due to the nature of the signal being composed of piecewise polynomials of degree zero. The differencing terms (derivatives) in the minimizing criterion are zero for the L_1 and L_0 criterion, except where the signal changes shape. This is not the case for the discrete-time version of the second derivative, so we have a slightly noisier signal. For the IRLS algorithm, as can be seen in the Appendix, we've computed the percent error for each iteration of the algorithm. For the L_1 and L_0 cases,

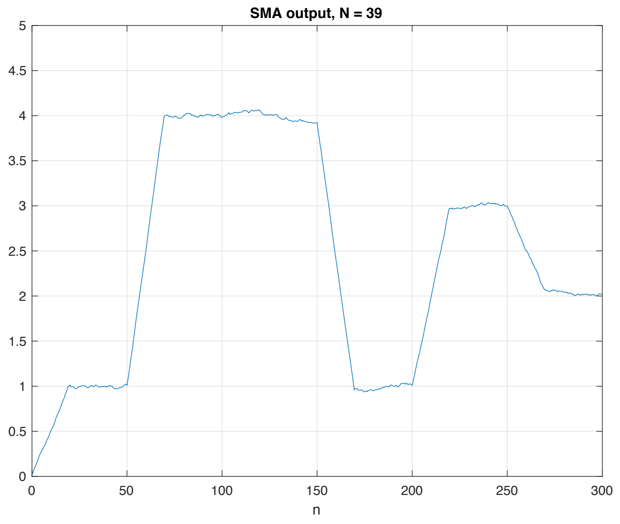




We can choose any number of iterations we choose to obtain a desired precision, but in these examples the error is asymptotic to zero after about 20 iterations.

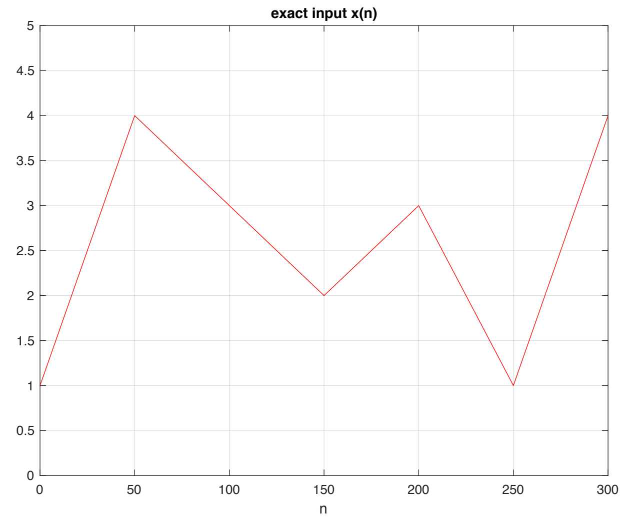
For comparison purposes, we'll show the output using an EMA and an equivalent SMA filter with N = 30 (which is one less than the flat-top portions to observe the transient behavior).

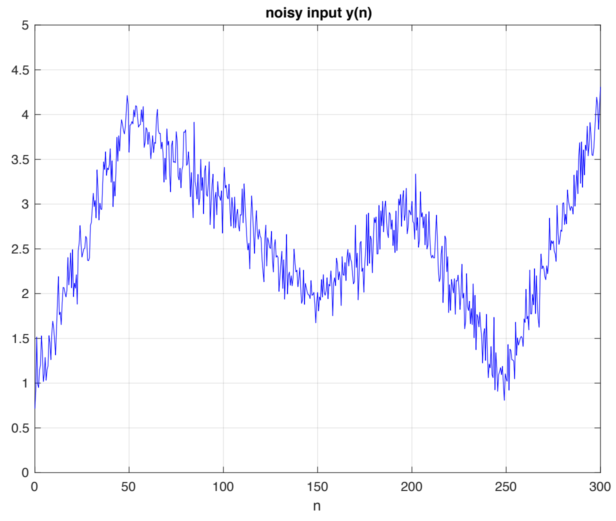




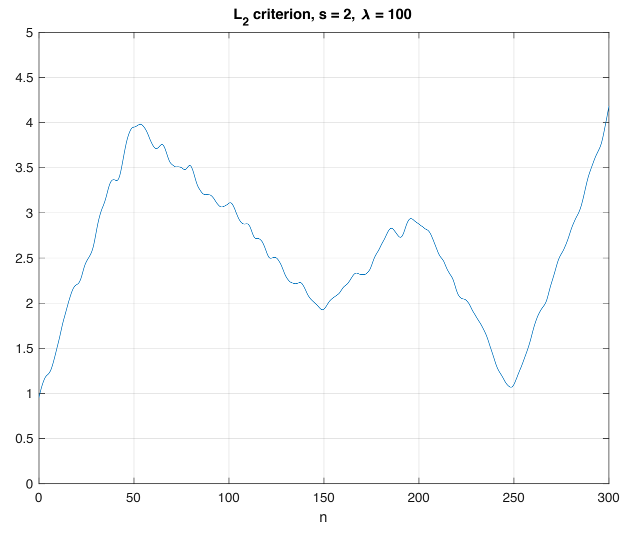
We can see that the Whittaker-Henderson methods obtain much better results.

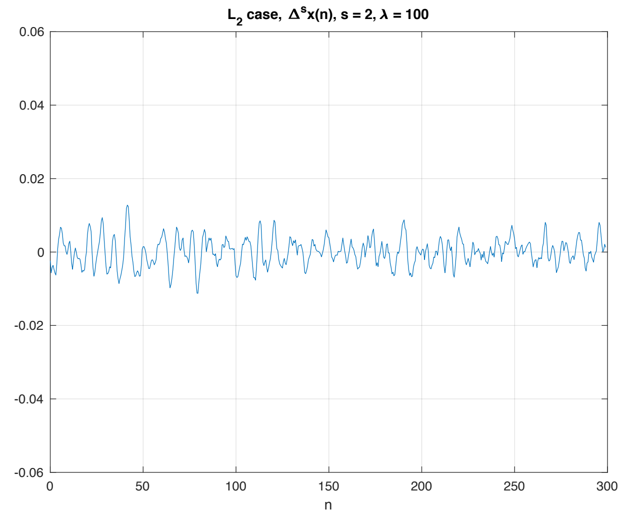
Now we'll repeat this procedure with a piecewise linear signal of the same duration with s = 2 and λ = 100.



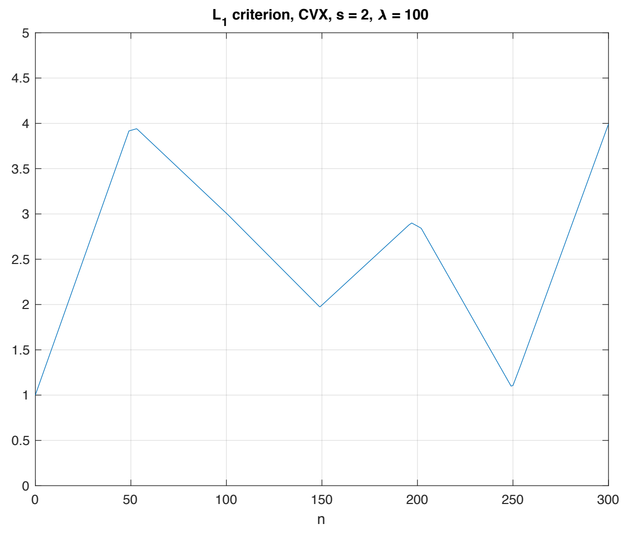


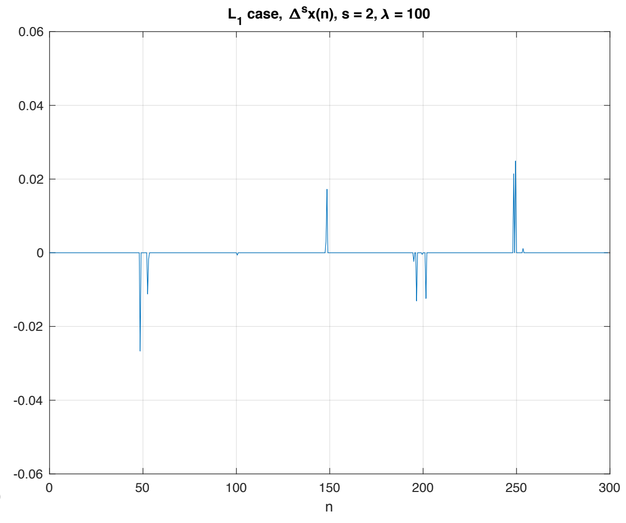
For the L_2 criterion we have,

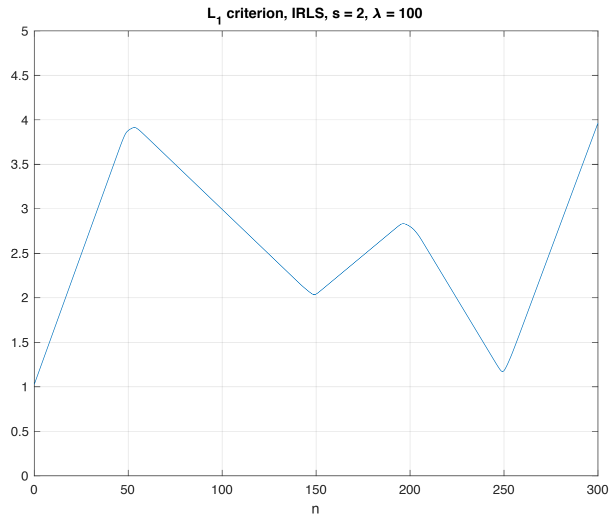


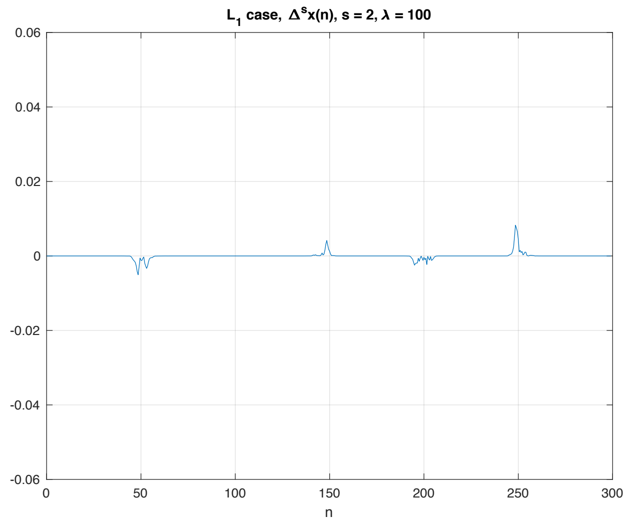


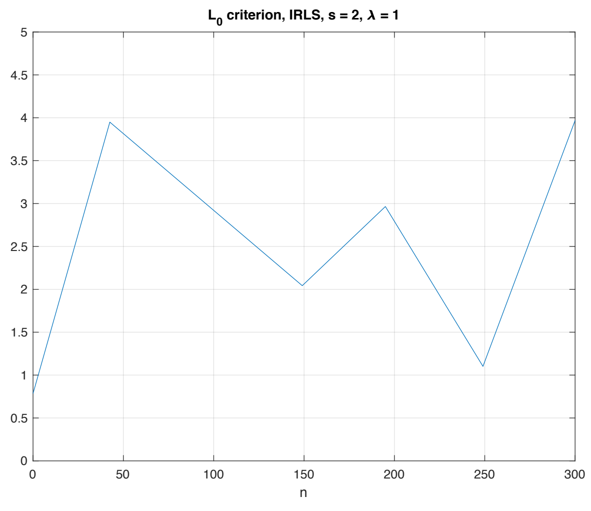
These results are similar, again because the L_2 criterion are best suited for polynomials of degree greater than one. For the L_1 and L_0 criterion, we have

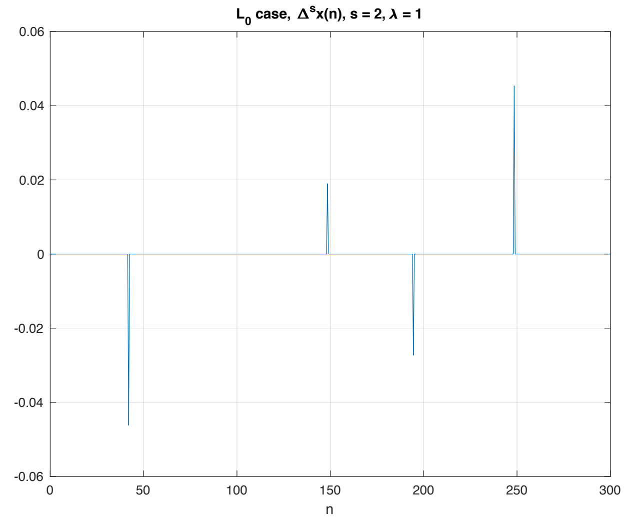

L_1 criterion, CVX, s = 2, λ = 100
n


L_1 case, $\Delta^s x(n)$, s = 2, λ = 100
n

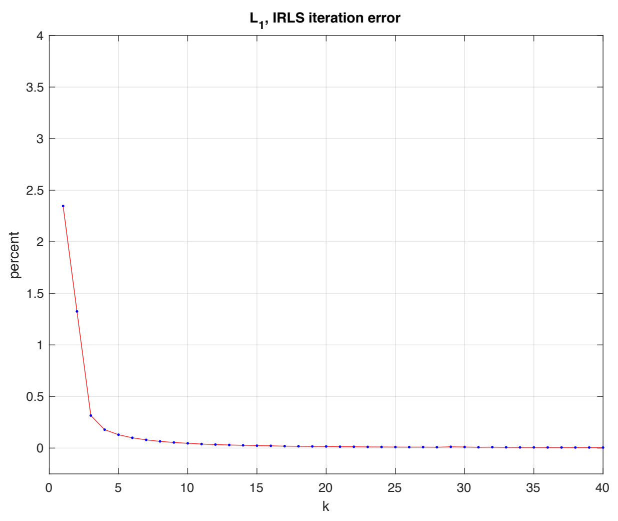

L_1 criterion, IRLS, s = 2, λ = 100
n

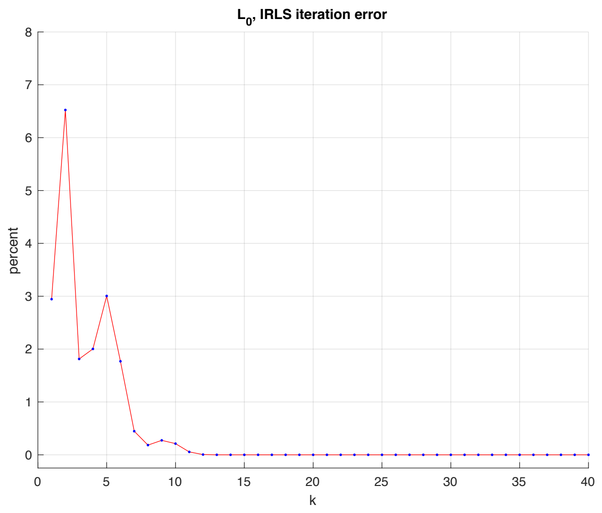

L_1 case, $\Delta^s x(n)$, s = 2, λ = 100
n


L_0 criterion, IRLS, s = 2, λ = 1
n

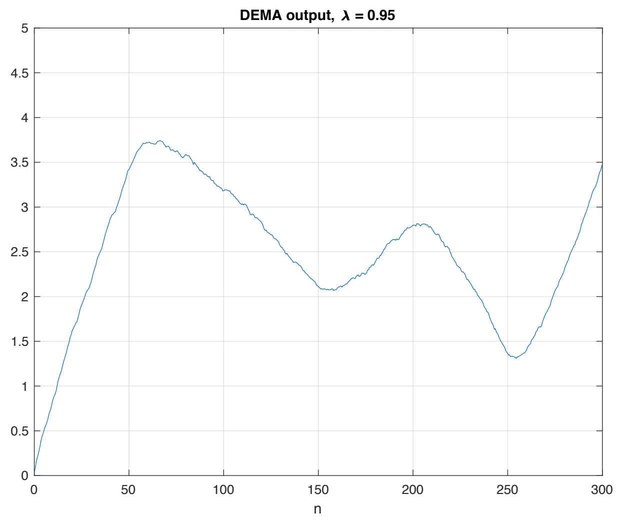

L_0 case, $\Delta^s x(n)$, s = 2, λ = 1
n

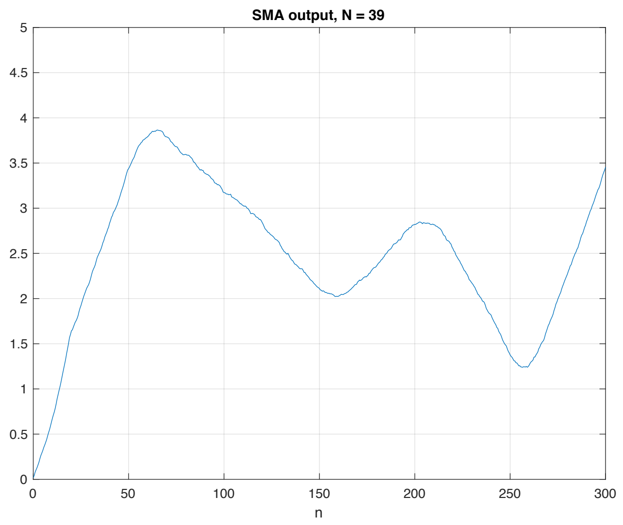
Comparing the iteration percentage errors,



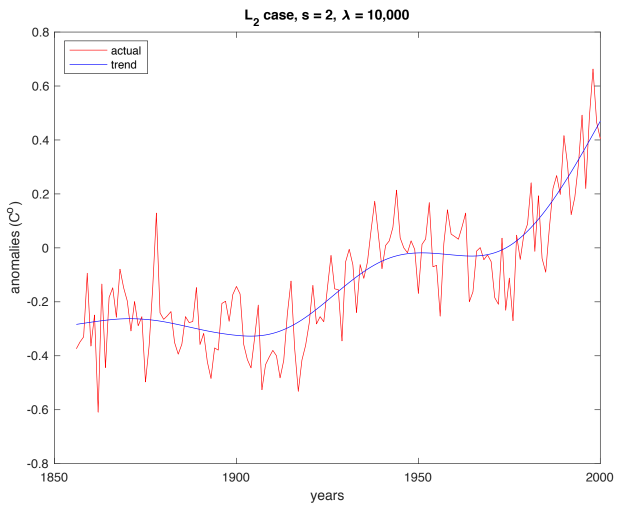


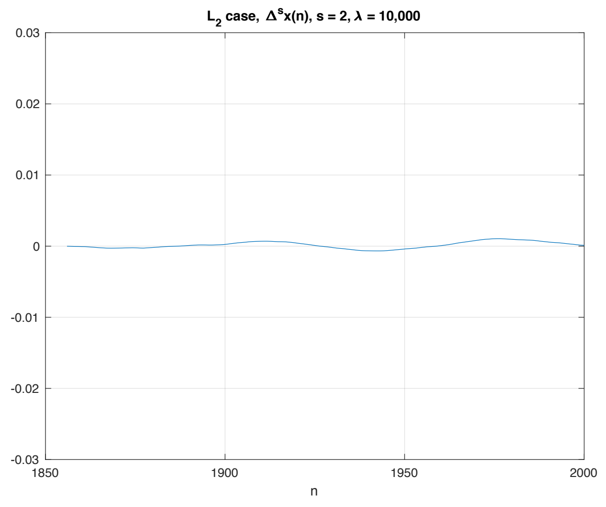
We'll compare these results to the output of a DEMA and an equivalent SMA.

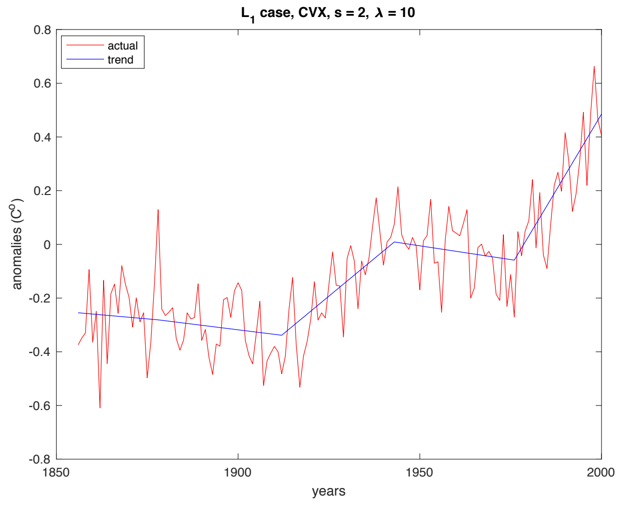


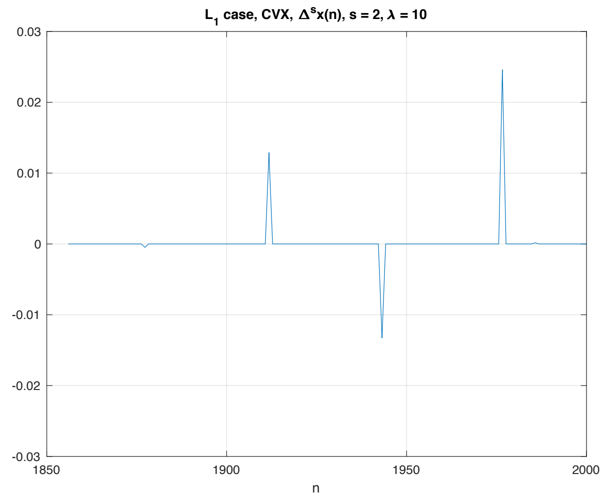


Finally, we'll look at a signal whose trend is nonlinear so we can see the higher performance of the L_2 criterion. Our data samples will be superimposed in the background to make the comparison easier.

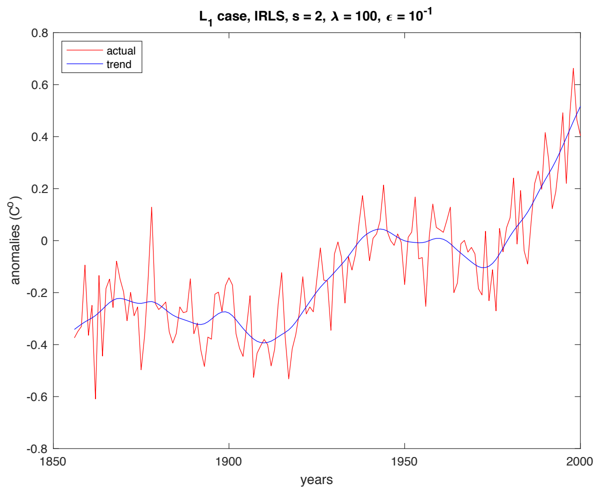


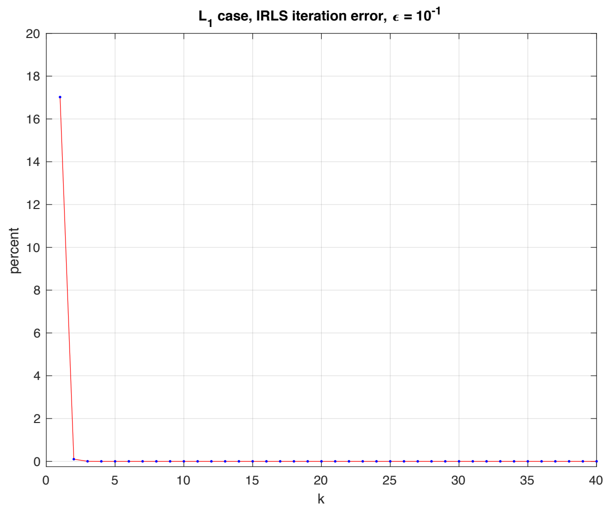




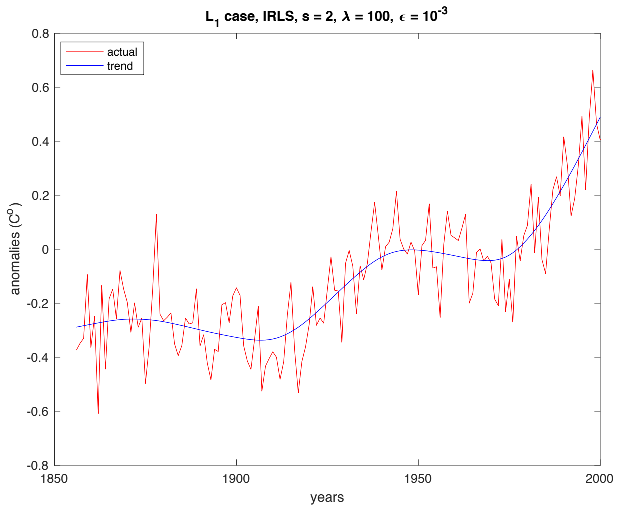


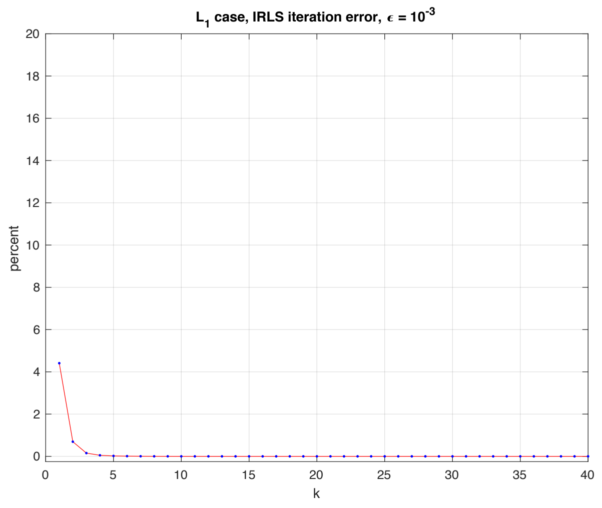
We can see that the L_2 criterion matches the data much more closely and reduces noise much better than in the previous examples, and, although its signal is not sparse, it is of minimal energy. Now, for the IRLS method, as mentioned before, choosing the right values of λ and ϵ involves some trial and error. The following results are for ϵ being a bit too large.



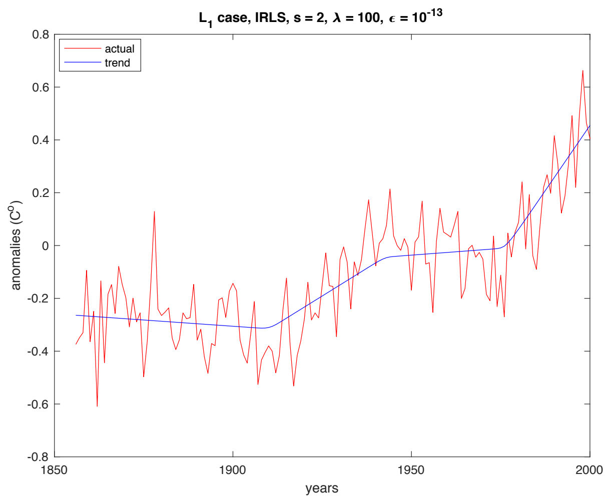


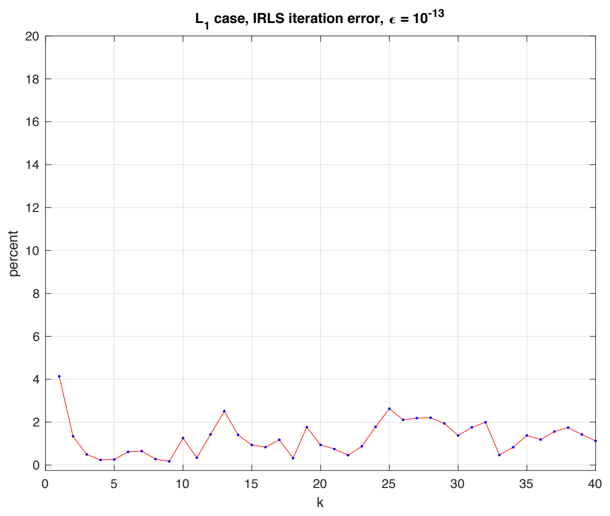
Decreasing the value of ϵ, we obtain,



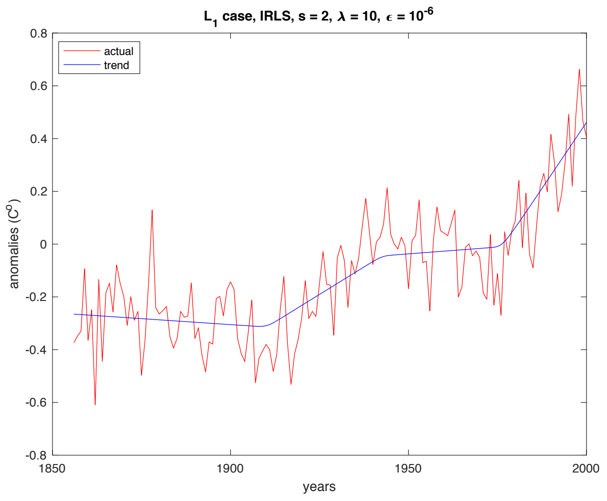


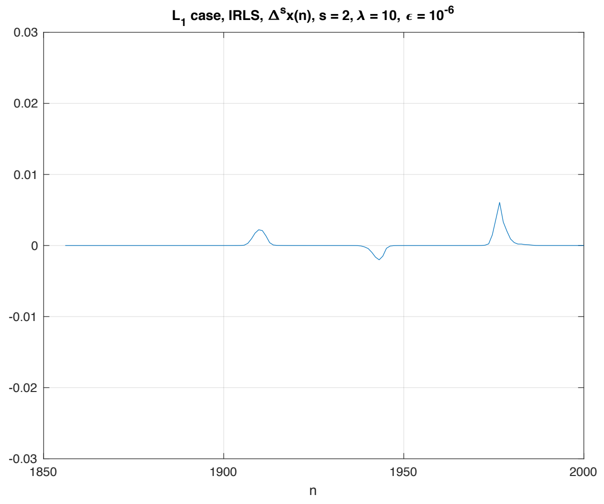
which is actually a desirable result, but it's sparsest for a piecewise linear function, as in the result obtained using the CVX package. So we'll go to the smallest value of ϵ possible to test the extreme.

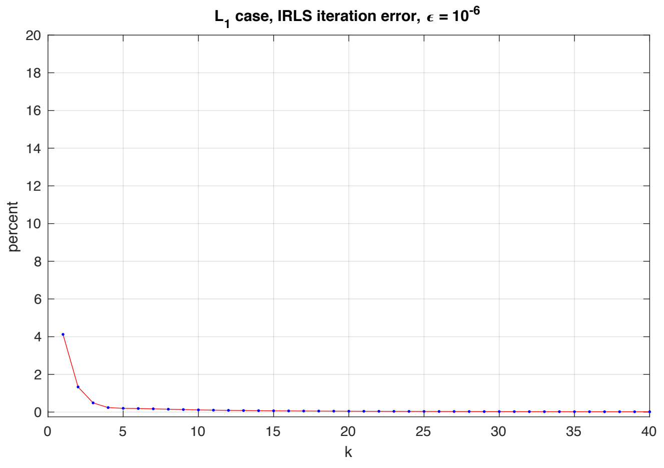




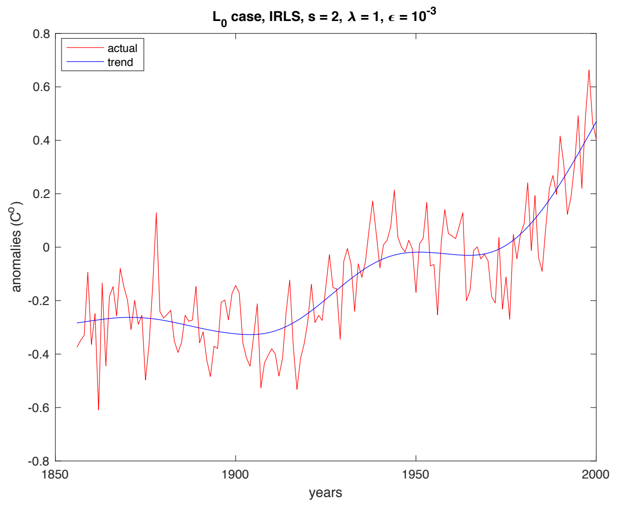
This gives us the desire result, but our iteration error does not converge, so our solution must involve a value of ϵ somewhere between these values. After more trial and error, we obtain the following results, which required decreasing λ as well.

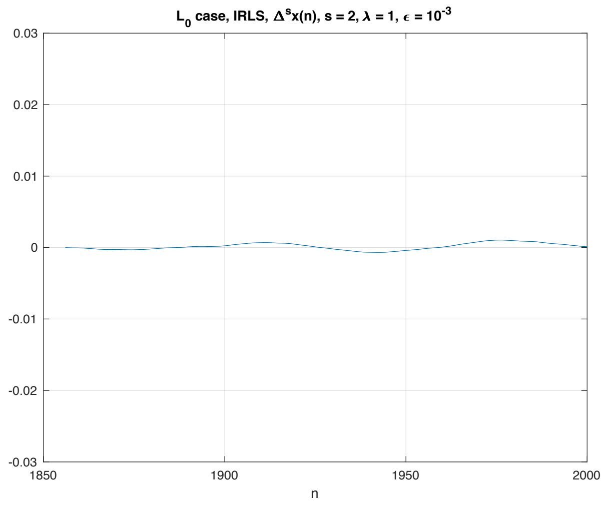




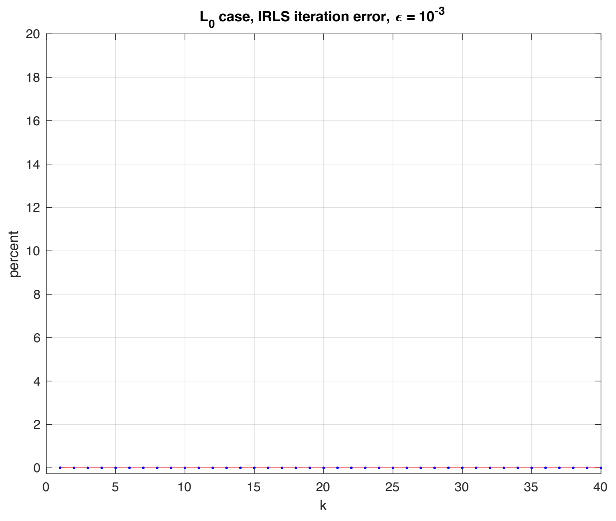


This gives us a result close to that of the CVX package; not quite as sparse, but an acceptable result. Now, repeating this procedure for the L_0 criterion we obtain the following results.





Here we settled for the solution that was not piecewise linear yet still had a minimal energy solution and a more than acceptable iteration percent error.



<u>References</u>

[0] Sophocles J. Orfanidis, DSP Design Project Resources, Rutgers University, 2019.

[1] S. J. Orfanidis, Applied Optimum Signal Processing, 2017 draft

[2] https://en.wikipedia.org/wiki/Condition_number#Matrices

Table of Contents

1a: Simulation Example 1a

```
% Load noisy version of flat-top signal
load yflat;
N = length(x);
n = 0:length(x) - 1;
% Plot exact input
close all
plot(n/2,x,'r');title('exact input x(n)');xlabel('n');grid on;
axis([0,300,0,5]);
% Plot noisy input
close all
plot(n/2,y,'b');title('noisy input y(n)');xlabel('n');grid on;
axis([0,300,0,5]);
```

1b: Simulation Example 1b

```
% number of backward-difference iterations (derivatives)
s = 1;
% regularization parameter
lambda = 5;
Ds = diff(speye(N),s);      % (N - s) x N, s-difference convolution
 matrix
x1b = (speye(N) + lambda*(Ds.')*Ds)\y;
Dsx1b = Ds*x1b;

% Plot results
nDsx = 0:length(Dsx1b)-1;
close all
plot(n/2,x1b);title('L_2 criterion, s = 1, \lambda = 5');xlabel('n');
```

```
grid on;axis([0,300,0,5]);

close all
plot(nDsx/2,Dsx1b);title('L_2 case, \Delta^sx(n), s = 1, \lambda =
 5');
xlabel('n');grid on;axis([0,300,-3, 3]);
```

1c: Using the CVX Package

```
cvx_begin;
    variable x1cCVX(N);
    minimize(sum_square(x1cCVX - y) + lambda*norm(Ds*x1cCVX,1));
cvx_end;

Dsx1cCVX = Ds*x1cCVX;
% Plot Results
close all
plot(n/2,x1cCVX);title('L_1 criterion, CVX, s = 1, \lambda = 5');
xlabel('n');grid on;axis([0,300,0,5]);
%
close all
plot(nDsx/2,Dsx1cCVX);title('L_1 case, \Delta^sx(n), s = 1, \lambda =
 5');
xlabel('n');grid on;axis([0,300,-3, 3]);
```

1c: Using the IRLS L1 Method

```
K = 40;      % Number of iterations
% Inital x is defined as x1b in part 1b
x1cIRLSL1 = x1b;
p = 1; % L1 norm
q = 2-p; % To define denominator of weighted L2 norm
eps = 1e-10;

for k = 1:K;
W1cIRLS = diag(1./(abs(Ds*x1cIRLSL1).^q + eps));
xtemp = (speye(N) + lambda*(Ds.')*W1cIRLS*Ds)\y;
PL1(k) = 100*norm((xtemp - x1cIRLSL1), 2)/norm(xtemp, 2);
x1cIRLSL1 = xtemp;
end

Dsx1cIRLSL1 = Ds*x1cIRLSL1;
% Plot Results
close all
plot(n/2,x1cIRLSL1);title('L_1 criterion, IRLS, s = 1, \lambda = 5');
xlabel('n');grid on;axis([0,300,0,5]);
%
close all
plot(nDsx/2,Dsx1cIRLSL1);title('L_1 case, \Delta^sx(n), s = 1, \lambda
 = 5');
xlabel('n');grid on;axis([0,300,-3, 3]);
%
```

```
k = 1:K;
close all
plot(k, PL1,'r',k,PL1,'b.'),title('L_1, IRLS iteration
 error');xlabel('k');
ylabel('percent'),grid on;axis([0,40,-0.25,8])
```

1c: Using the IRLS L0 Method

```
K = 40;     % Number of iterations
% Inital x is defined as x1b in part 1b
x1cIRLSL0 = x1b;
p = 0; % L0 norm
q = 2-p; % To define denominator of weighted L2 norm
eps = 1e-10;

for k = 1:K;
W1cIRLS = diag(1./(abs(Ds*x1cIRLSL0).^q + eps));
xtemp = (speye(N) + lambda*(Ds.')*W1cIRLS*Ds)\y;
PL0(k) = 100*norm((xtemp - x1cIRLSL0), 2)/norm(xtemp, 2);
x1cIRLSL0 = xtemp;
end

Dsx1cIRLSL0 = Ds*x1cIRLSL0;
% Plot Results
close all
plot(n/2,x1cIRLSL0);title('L_0 criterion, IRLS, s = 1, \lambda = 5');
xlabel('n');grid on;axis([0,300,0,5]);
%
close all
plot(nDsx/2,Dsx1cIRLSL0);title('L_0 case, \Delta^sx(n), s = 1, \lambda
 = 5');
xlabel('n');grid on;axis([0,300,-3, 3]);
%
k = 1:K;
close all
plot(k, PL0,'r',k,PL0,'b.'),title('L_0, IRLS iteration
 error');xlabel('k');
ylabel('percent'),grid on;axis([0,40,-0.25,8])
```

1d: Comparing the EMA and SMA

```
% EMA
alpha = 0.95;
b = 1-alpha;
a = [1,-alpha];
x1cEMA = filter(b,a,y);
% Plot Results
close all
plot(n/2,x1cEMA),title('EMA output, \lambda =
 0.95');xlabel('n');grid on;
axis([0,300,0,5]);

% SMA
```

```
L = round((1 + alpha)/(1 - alpha));
bSMA = (1/L)*ones(L,1);
% Plot Results
x1cSMA = filter(bSMA,1,y);
close all
plot(n/2,x1cSMA),title('SMA output, N = 39');xlabel('n');grid on;
axis([0,300,0,5]);
```

2a: Simulation Example 2a

```
% Load noisy piecewise linear signal
load ylin;
N = length(x);
n = 0:length(x) - 1;
% Plot exact input
close all
plot(n/2,x,'r');title('exact input x(n)');xlabel('n');grid on;
axis([0,300,0,5]);
% Plot noisy input
close all
plot(n/2,y,'b');title('noisy input y(n)');xlabel('n');grid on;
axis([0,300,0,5]);
```

2b: Simulation Example 2b

```
% number of backward-difference iterations (derivatives)
s = 2;
% regularization parameter
lambda = 100;
Ds = diff(speye(N),s);      % (N - s) x N, s-difference convolution
 matrix
x2b = (speye(N) + lambda*(Ds.')*Ds)\y;
Dsx2b = Ds*x2b;
% Plot Results
nDsx = 0:length(Dsx2b)-1;
close all
plot(n/2,x2b);title('L_2 criterion, s = 2, \lambda =
 100');xlabel('n');
grid on;axis([0,300,0,5]);
%
close all
plot(nDsx/2,Dsx2b);title('L_2 case, \Delta^sx(n), s = 2, \lambda =
 100');
xlabel('n');grid on;axis([0,300,-0.06, 0.06]);
```

2c: Using the CVX Package

```
cvx_begin;
    variable x2cCVX(N);
    minimize(sum_square(x2cCVX - y) + lambda*norm(Ds*x2cCVX,1));
cvx_end;
```

```
Dsx2cCVX = Ds*x2cCVX;
% Plot Results
close all
plot(n/2,x2cCVX);title('L_1 criterion, CVX, s = 2, \lambda = 100');
xlabel('n');grid on;axis([0,300,0,5]);
%
close all
plot(nDsx/2,Dsx2cCVX);title('L_1 case, \Delta^sx(n), s = 2, \lambda =
 100');
xlabel('n');grid on;axis([0,300,-0.06, 0.06]);
```

2c: Using the IRLS L1 Method

```
K = 40;     % Number of iterations
% Inital x is defined as x2b in part 2b
x2cIRLSL1 = x2b;
p = 1; % L1 norm
q = 2-p; % To define denominator of weighted L2 norm
eps = 1e-10;

for k = 1:K
W2cIRLS = diag(1./(abs(Ds*x2cIRLSL1).^q + eps));
xtemp = (speye(N) + lambda*(Ds.')*W2cIRLS*Ds)\y;
PL1(k) = 100*norm((xtemp - x2cIRLSL1), 2)/norm(xtemp, 2);
x2cIRLSL1 = xtemp;
end

Dsx2cIRLSL1 = Ds*x2cIRLSL1;
% Plot Results
close all
plot(n/2,x2cIRLSL1);title('L_1 criterion, IRLS, s = 2, \lambda =
 100');
xlabel('n');grid on;axis([0,300,0,5]);
%
close all
plot(nDsx/2,Dsx2cIRLSL1);title('L_1 case, \Delta^sx(n), s = 2, \lambda
 = 100');
xlabel('n');grid on;axis([0,300,-0.06, 0.06]);
%
k = 1:K;
close all
plot(k, PL1,'r',k,PL1,'b.'),title('L_1, IRLS iteration
 error');xlabel('k');
ylabel('percent'),grid on;axis([0,40,-0.25,4])
```

2c: Using the IRLS L0 Method

```
lambda = 1;
K = 40;     % Number of iterations
% Inital x is defined as x1b in part 1b
x2cIRLSL0 = x2b;
p = 0; % L0 norm
q = 2-p; % To define denominator of weighted L2 norm
```

```
eps = 1e-10;

for k = 1:K
W2cIRLS = diag(1./(abs(Ds*x2cIRLSL0).^q + eps));
xtemp = (speye(N) + lambda*(Ds.')*W2cIRLS*Ds)\y;
PL0(k) = 100*norm((xtemp - x2cIRLSL0), 2)/norm(x2cIRLSL0, 2);
x2cIRLSL0 = xtemp;
end

Dsx2cIRLSL0 = Ds*x2cIRLSL0;
% Plot Results
close all
plot(n/2,x2cIRLSL0);title('L_0 criterion, IRLS, s = 2, \lambda = 1');
xlabel('n');grid on;axis([0,300,0,5]);
%
close all
plot(nDsx/2,Dsx2cIRLSL0);title('L_0 case, \Delta^sx(n), s = 2, \lambda
 = 1');
xlabel('n');grid on;axis([0,300,-0.06, 0.06]);
%
k = 1:K;
close all
plot(k, PL0,'r',k,PL0,'b.'),title('L_0, IRLS iteration
 error');xlabel('k');
ylabel('percent'),grid on;axis([0,40,-0.25,8])
```

2d: Comparing the DEMA and SMA

```
% DEMA
[a,b,a1,a2,cinit] = dema(y,N); % Using output of first EMA as input to
 second EMA at time n = N - 1
% Plot Results
close all
%plot(n/2,x2cDEMA),title('DEMA output, \lambda =
 0.95');xlabel('n');grid on;
axis([0,300,0,5]);

% SMA
L = round((1 + alpha)/(1 - alpha));
bSMA = (1/L)*ones(L,1);
x2cSMA = filter(bSMA,1,y);
% Plot Results
close all
plot(n/2,x2cSMA),title('SMA output, N = 39');xlabel('n');grid on;
axis([0,300,0,5]);
```

3: Global Warming Trends

```
% Load global warming data
load tavenh2v.dat
% Extract dates
dates = tavenh2v(:,1); avg = tavenh2v(:,end);
N = length(avg);
```

```
n = 0:length(avg) - 1;
```

3a: Using the Standard L2 Method

```
s = 2; lambda = 10000;
Ds = diff(speye(N),s);      % (N - s) x N, s-difference convolution
 matrix
x3 = (speye(N) + lambda*(Ds.')*Ds)\avg;
Dsx3 = Ds*x3;
nDsx = linspace(dates(1),dates(end),length(Dsx3));
% Plot Results
close all
plot(dates,avg,'r',dates,x3,'b');title('L_2 case, s = 2, \lambda =
 10,000');
xlabel('years');axis([1850,2000,-0.8,0.8]);legend('actual','trend');
ylabel('anomalies (C^o)');
%
close all
plot(nDsx,Dsx3);title('L_2 case, \Delta^sx(n), s = 2, \lambda =
 10,000');
xlabel('n');grid on;axis([1850,2000,-0.03, 0.03]);
```

3b: Using the CVX Package, L1 Method

```
lambda = 10;
cvx_begin;
    variable x3CVX(N);
    minimize(sum_square(x3CVX - avg) + lambda*norm(Ds*x3CVX,1));
cvx_end;

Dsx3CVX = Ds*x3CVX;
nDsx = linspace(dates(1),dates(end),length(Dsx3CVX));
% Plot Results
close all
plot(dates,avg,'r',dates,x3CVX,'b');title('L_1 case, CVX, s = 2,
 \lambda = 10');
xlabel('years');axis([1850,2000,-0.8,0.8]);legend('actual','trend');
ylabel('anomalies (C^o)');
%
close all
plot(nDsx,Dsx3CVX);title('L_1 case, CVX, \Delta^sx(n), s = 2, \lambda
 = 10');
xlabel('n');grid on;axis([1850,2000,-0.03, 0.03]);
```

3c: Using the IRLS L1 Method

```
lambda = 10;
K = 40;     % Number of iterations
% Inital x is defined as x3 in the beginning of part 3
x3IRLSL1 = x3;
p = 1; % L1 norm
q = 2-p; % To define denominator of weighted L2 norm
```

```
eps = 1e-6;

for k = 1:K
W3IRLS = diag(1./(abs(Ds*x3IRLSL1).^q + eps));
xtemp = (speye(N) + lambda*(Ds.')*W3IRLS*Ds)\avg;
PL1(k) = 100*norm((xtemp - x3IRLSL1), 2)/norm(xtemp, 2);
x3IRLSL1 = xtemp;
end

Dsx3IRLSL1 = Ds*x3IRLSL1;
% Plot Results
close all
plot(dates,avg,'r',dates,x3IRLSL1,'b');title('L_1 case, IRLS, s = 2,
 \lambda = 10, \epsilon = 10^{-6}');
xlabel('years');axis([1850,2000,-0.8,0.8]);legend('actual','trend');
ylabel('anomalies (C^o)');
%
close all
plot(nDsx,Dsx3IRLSL1);title('L_1 case, IRLS, \Delta^sx(n), s = 2,
 \lambda = 10, \epsilon = 10^{-6}');
xlabel('n');grid on;axis([1850,2000,-0.03, 0.03]);
%
k = 1:K;
close all
plot(k, PL1,'r',k,PL1,'b.'),title('L_1 case, IRLS iteration error,
 \epsilon = 10^{-6}');xlabel('k');
ylabel('percent'),grid on;axis([0,40,-0.25, 20]);
```

3: Using the IRLS L0 Method

```
lamda = 1;
K = 40;     % Number of iterations
% Inital x is defined as x3 in the beginning of part 3
x3IRLSL0 = x3;
p = 0; % L0 norm
q = 2-p; % To define denominator of weighted L2 norm
eps = 1e-3;

for k = 1:K
W3IRLS = diag(1./(abs(Ds*x3IRLSL0).^q + eps));
xtemp = (speye(N) + lambda*(Ds.')*W3IRLS*Ds)\avg;
PL0(k) = 100*norm((xtemp - x3IRLSL0), 2)/norm(xtemp, 2);
x3IRLSL0 = xtemp;
end

Dsx3IRLSL0 = Ds*x3IRLSL0;
% Plot Results
close all
plot(dates,avg,'r',dates,x3IRLSL0,'b');title('L_0 case, IRLS, s = 2,
 \lambda = 1, \epsilon = 10^{-3}');
xlabel('years');axis([1850,2000,-0.8,0.8]);legend('actual','trend');
ylabel('anomalies (C^o)');
%
```

```
close all
plot(nDsx,Dsx3IRLSL0);title('L_0 case, IRLS, \Delta^sx(n), s = 2,
 \lambda = 1, \epsilon = 10^{-3}');
xlabel('n');grid on;axis([1850,2000,-0.03, 0.03]);
%
k = 1:K;
close all
plot(k, PL0,'r',k,PL0,'b.'),title('L_0 case, IRLS iteration error,
 \epsilon = 10^{-3}');xlabel('k');
ylabel('percent'),grid on;axis([0,40,-0.25, 20]);
```

Published with MATLAB® R2017b