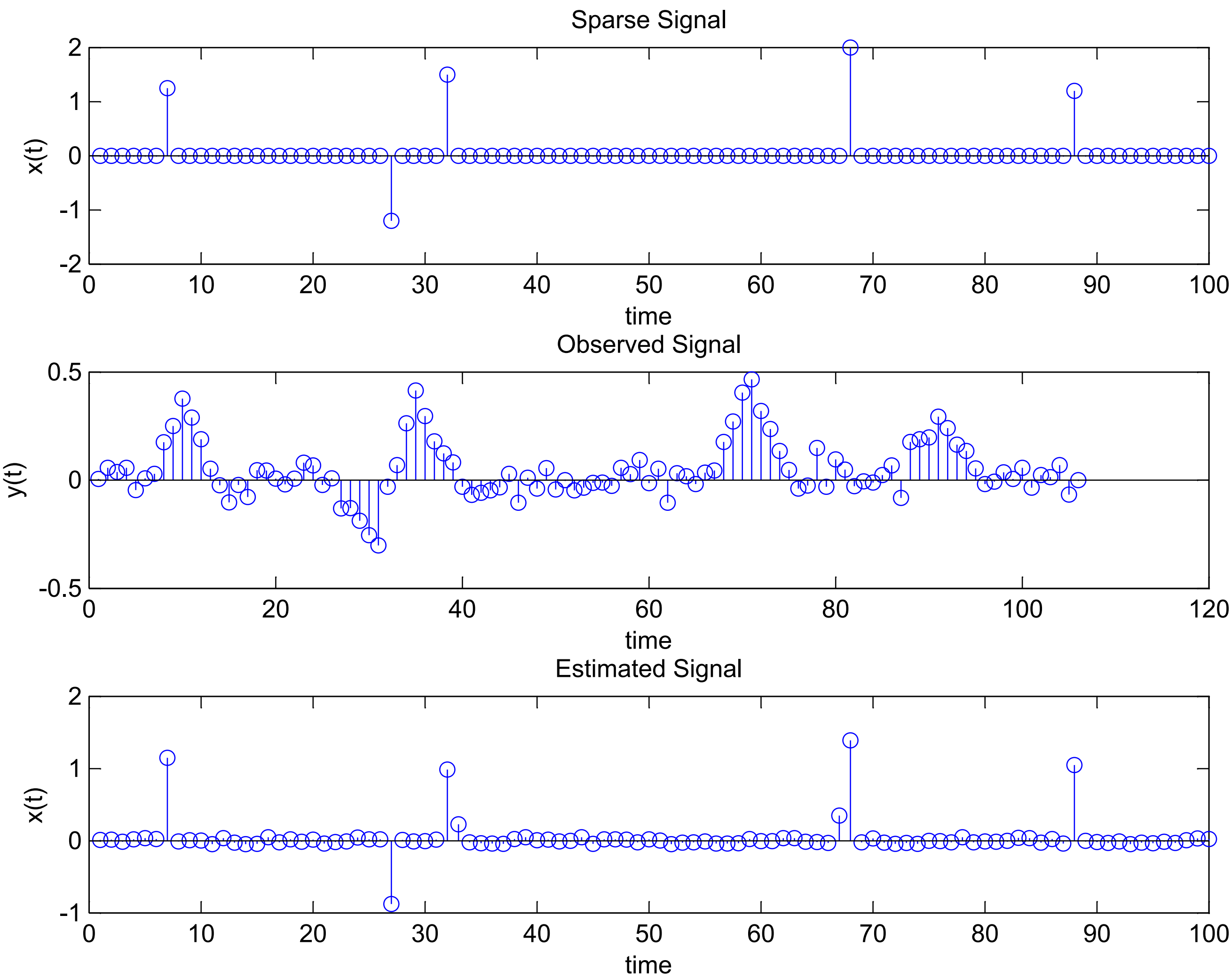
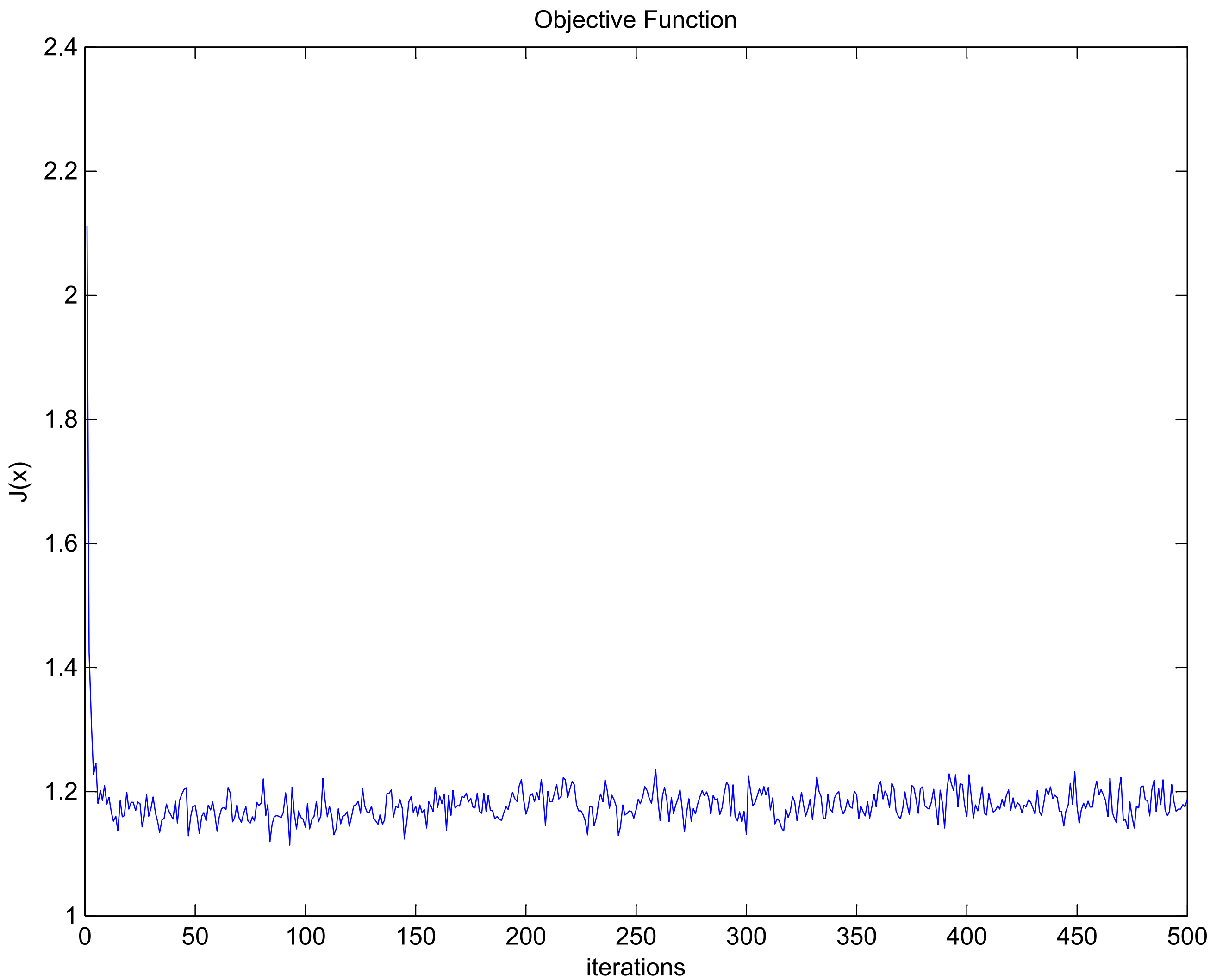
```
>> % Summary Cross Section
>> % Main goal: building/applying fast algorithms for ECG signals approximation
>> % Let's consider an iterative soft (shrinkage) - thresholding algorithm (ISTA)
>> % when assuming an observed signal y = Hx + n is known
>> % then by minimizing an objective function J(x)
>> % J(x) = NORM2( y - Hx )^2 + lambda*NORM1(x)
>> % by applying an iterative rule soft() one can find the best matching x
>> % xk+1 = soft( xk + 2/alphaH'(y-Hxk), lambda/(2alpha) )
>> function [x, J ] = ista( y, H, lambda, alpha, Nit )

% [x, J] = ista(y, H, lambda, alpha, Nit)
% L1-regularized signal restoration using the iterated
% soft-thresholding algorithm (ISTA)
% Minimizes J(x) = norm2(y-H*x)^2 + lambda*norm1(x)
% INPUT
% y - observed signal
% H - matrix or operator
% lambda - regularization parameter
% alpha - need alpha >= max(eig(H'*H))
% Nit - number of iterations
% OUTPUT
% x - result of deconvolution
% J - objective function

J = zeros(1, Nit); % Objective function
x = 0*H'*y; % Initialize x
T = lambda/(2*alpha);

for k = 1: Nit
    Hx = H*x;
    J(k) = sum(abs(Hx(:)-y(:)).^2) + lambda*sum(abs(x(:)));
    x = soft(x + (H'*(y - Hx))/alpha, T);
end
end

function [x] = soft(y, T)
x = y - T*sign(y);
end
 function [x, J ] = ista( y, H, lambda, alpha, Nit )
 |
Error: Function definitions are not permitted in this context.

>>
```

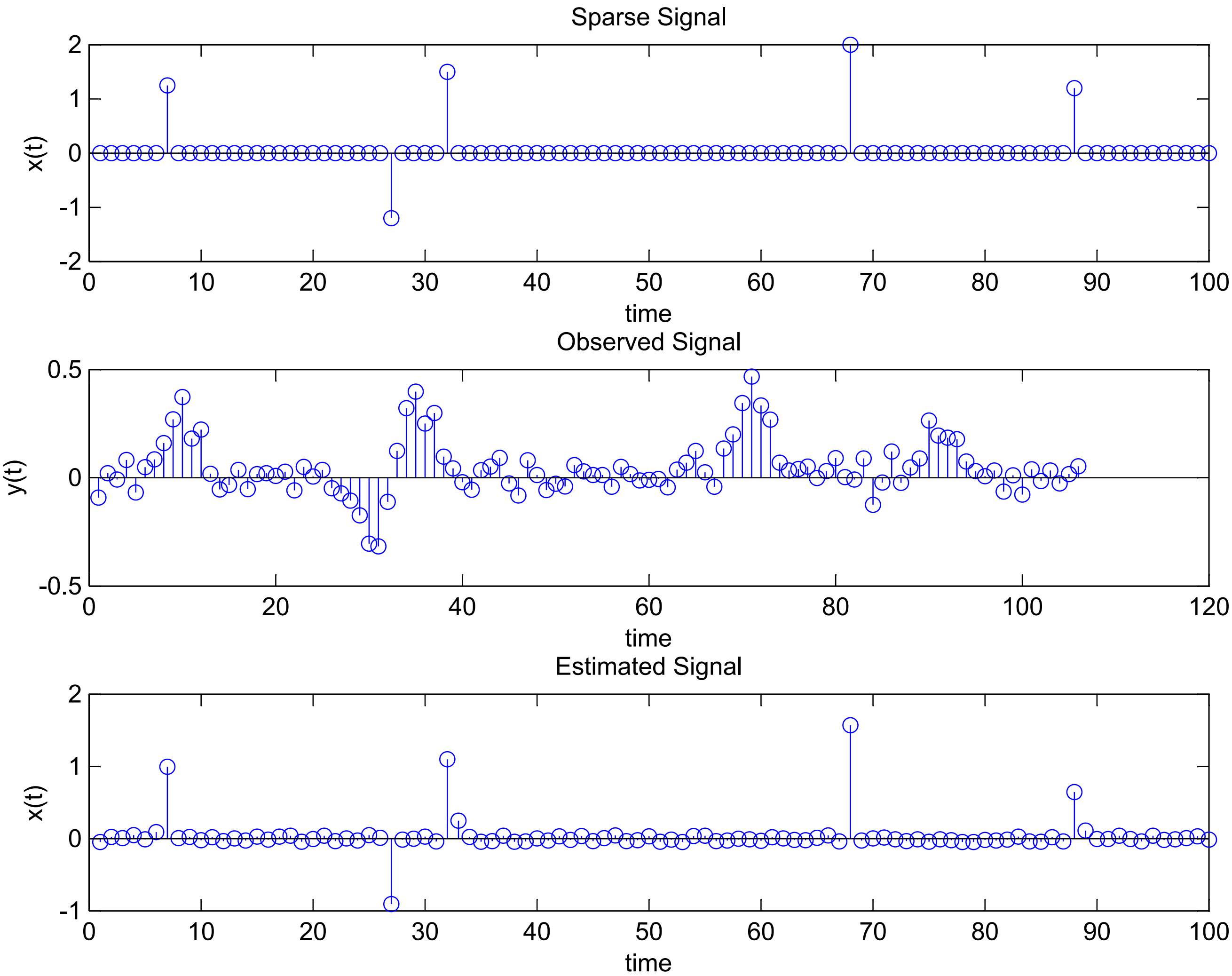
```
>> % Using the following script we can observe the outcoming results of ISTA
>> clc; clear all; close all;

%-----------Create-sparse-signal------------------------------------------------%
x = zeros(100, 1);
x(7) = 1.25; x(27) = -1.2;
x(32) = 1.5; x(68) = 2;
x(88) = 1.2;
%-----------------------------------------------------------------------------------%

%-----------Define-Parameters-------------------------------------------------%
h = [1 2 3 4 3 2 1]/16; % impulse response
N = 100; % num of samples of x
H = convmtx(h', N); % convolution matrix
lambda = 0.1; alpha = 1; % convergence parameters
Nit = 500; % num of iterations
%-----------------------------------------------------------------------------------%

%------------Observed-signal-with-noise---------------------------------------%
n = 0.05*randn(106, 1);
y = H*x + n;
%-----------------------------------------------------------------------------------%

% Apply ISTA
[x_est, J] = ista(y, H, lambda, alpha, Nit);


%-------------ISTA Results Printout--------------------------------------------%
figure(1)
% plot sparse signal
subplot(311); stem(x);
xlabel('time'); ylabel('x(t)');
title('Sparse Signal');

% plot observed signal
subplot(312); stem(y);
xlabel('time'); ylabel('y(t)');
title('Observed Signal');

% plot estimated signal
subplot(313); stem(x_est);
xlabel('time'); ylabel('x(t)');
title('Estimated Signal');


figure(2);
% error versus num of iteration
plot(J); xlabel('iterations'); ylabel('J(x)');
title('Objective Function');
%-----------------------------------------------------------------------------------%
```


Sparse Signal
x(t)
2
1
0
-1
-2
0
10
20
30
40
50
60
70
80
90
100
time
Observed Signal
y(t)
0.5
0
-0.5
0
20
40
60
80
100
120
time
Estimated Signal
x(t)
2
1
0
-1
0
10
20
30
40
50
60
70
80
90
100
time

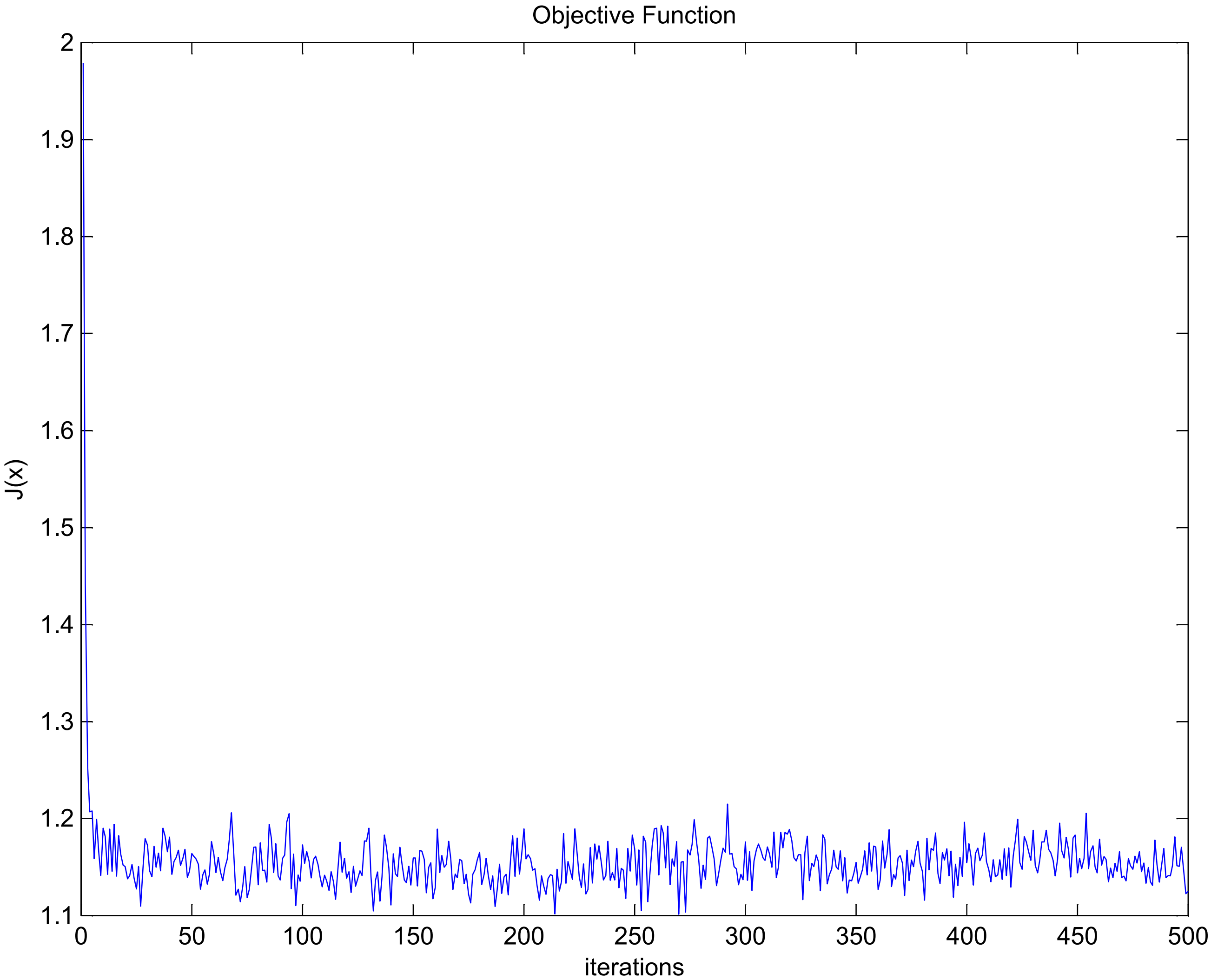

Objective Function
J(x)
iterations
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0
50
100
150
200
250
300
350
400
450
500

```
>> % to avoid large matrices computations one could be implemented using function defin
>> function [x,J] = ista_fns(y, H, Ht, lambda, alpha, Nit)
% [x, J] = ista_fns(y, H, lambda, alpha, Nit)

J = zeros(1, Nit); % Objective function
x = 0*Ht(y); % Initialize x
T = lambda/(2*alpha);

for k = 1:Nit
    Hx = H(x);
    J(k) = sum(abs(Hx(:)-y(:)).^2) + lambda*sum(abs(x(:)));
    x = soft(x + (Ht(y - Hx))/alpha, T);
end

end

function [x] = soft(y, T)
x = y - T*sign(y);
end
```

```
>> % using the following script to test ista_fns
>> clc; clear all; close all;

%-----------Create-sparse-signal-------------------------------------------------%
x = zeros(100, 1);
x(7) = 1.25; x(27) = -1.2;
x(32) = 1.5; x(68) = 2;
x(88) = 1.2;
%--------------------------------------------------------------------------------%

%-----------Define-Parameters----------------------------------------------------%
h = [1 2 3 4 3 2 1]/16; % impulse response
N = 100; % num samples of x
lambda = 0.1; alpha = 1; % convergence parameters
Nit = 500; % num of iterations
H = @(x) conv(h, x); %function implementation instead of matrices
Ht = @(y) convt(h, y);
%--------------------------------------------------------------------------------%

%------------Observed-signal-with-noise--------------------------------------%
n = 0.05*randn(106, 1);
y = H(x) + n;
%--------------------------------------------------------------------------------%

% Apply ISTA_FNS
[x_est, J] = ista_fns(y, H, Ht, lambda, alpha, Nit);


%------------ISTA_FNS-Results-Printout---------------------------------------%
figure(1)
% plot sparse signal
subplot(311); stem(x);
xlabel('time'); ylabel('x(t)');
title('Sparse Signal');

% plot observed signal
subplot(312); stem(y);
xlabel('time'); ylabel('y(t)');
title('Observed Signal');

% plot estimated signal
subplot(313); stem(x_est);
xlabel('time'); ylabel('x(t)');
title('Estimated Signal');


figure(2);
% error versus num of iteration
plot(J); xlabel('iterations'); ylabel('J(x)');
title('Objective Function');
%--------------------------------------------------------------------------------%
```



Sparse Signal
x(t)
2
1
0
-1
-2
0
10
20
30
40
50
60
70
80
90
100
time
Observed Signal
y(t)
0.5
0
-0.5
0
20
40
60
80
100
120
time
Estimated Signal
x(t)
2
1
0
-1
0
10
20
30
40
50
60
70
80
90
100
time


Objective Function
J(x)
iterations
2
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
0
50
100
150
200
250
300
350
400
450
500

```
>> % now let's try to match ISTA to our ECG signal
>> % to do that we assume h be of gaussian shape
>> % for a visual example define h:
>> h = [1 2 1]

h =

     1     2     1

>> % normalizing h:
>> h = h./sum(h)

h =

    0.2500    0.5000    0.2500

>> % number of samples in h
>> L = length(h)

L =

     3

>> % length of the observed signal y
>> % (also num of rows in convolution matrix)
>> Ny = 2*L

Ny =

     6

>> y = 1:Ny

y =

     1     2     3     4     5     6

>> H = convmtx(h, Ny)

H =

    0.2500    0.5000    0.2500         0         0         0         0         0
         0    0.2500    0.5000    0.2500         0         0         0         0
         0         0    0.2500    0.5000    0.2500         0         0         0
         0         0         0    0.2500    0.5000    0.2500         0         0
         0         0         0         0    0.2500    0.5000    0.2500         0
         0         0         0         0         0    0.2500    0.5000    0.2500

>> % guessing vector x
>> Nx = L + Ny - 1

Nx =

     8

>> x = 1:Nx
```

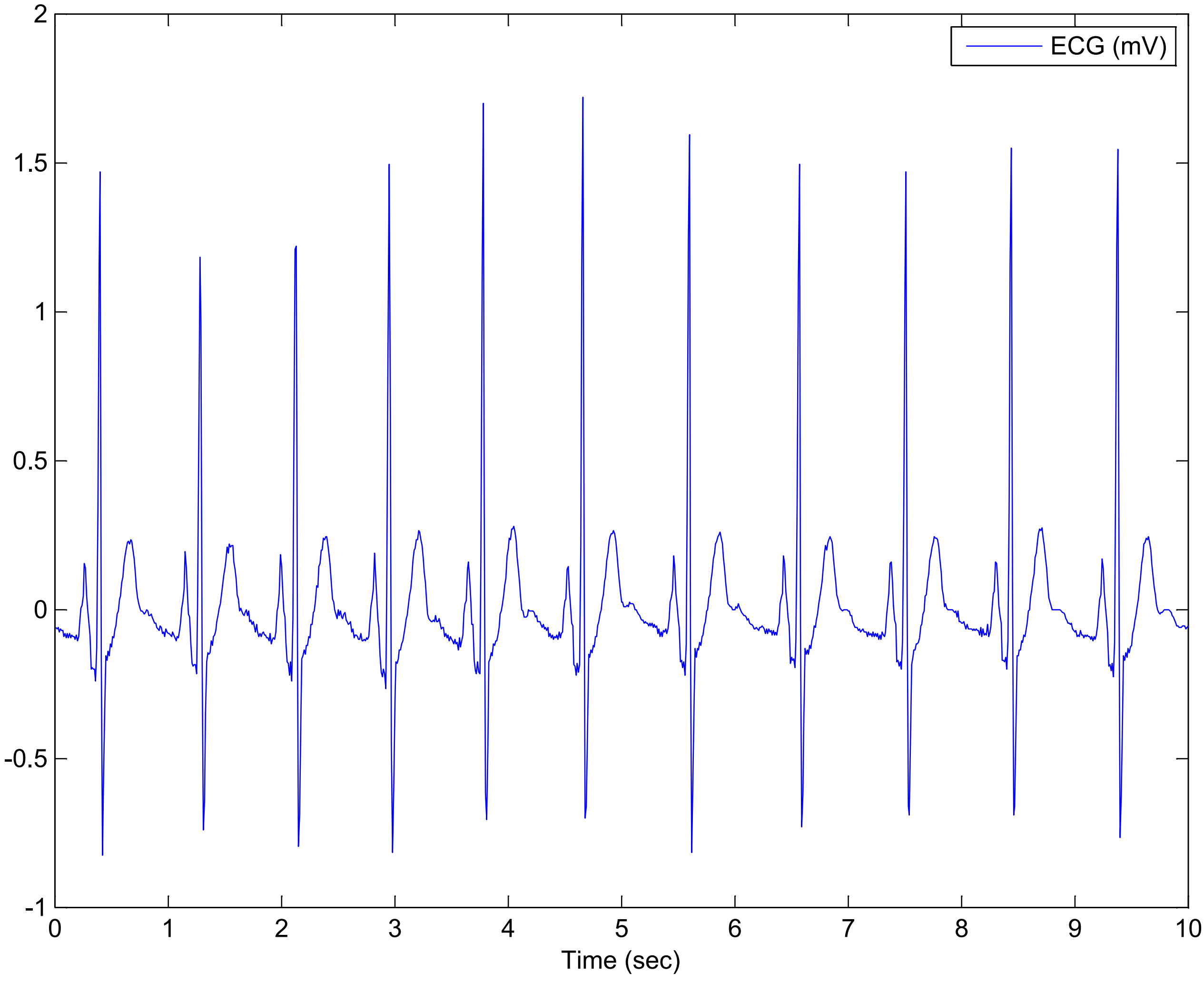
```
x =

     1     2     3     4     5     6     7     8

>> % Hx_mtx representing each column as h vector shifted in time
>> % and multiplied by the corresponding value of x
>> Hx_mtx = H*diag(x)

Hx_mtx =

    0.2500    1.0000    0.7500         0         0         0         0         0
         0    0.5000    1.5000    1.0000         0         0         0         0
         0         0    0.7500    2.0000    1.2500         0         0         0
         0         0         0    1.0000    2.5000    1.5000         0         0
         0         0         0         0    1.2500    3.0000    1.7500         0
         0         0         0         0         0    1.5000    3.5000    2.0000

>> % sum of the columns of the matrix above describes y
>> % as combinations of shifted gaussians of different amplitudes
>> Hx = H*x(:)

Hx =

     2
     3
     4
     5
     6
     7

>> % define corresponding transpose matrix
>> Ht = H'

Ht =

    0.2500         0         0         0         0         0
    0.5000    0.2500         0         0         0         0
    0.2500    0.5000    0.2500         0         0         0
         0    0.2500    0.5000    0.2500         0         0
         0         0    0.2500    0.5000    0.2500         0
         0         0         0    0.2500    0.5000    0.2500
         0         0         0         0    0.2500    0.5000
         0         0         0         0         0    0.2500

>> Hty_mtx = Ht*diag(y)

Hty_mtx =

    0.2500         0         0         0         0         0
    0.5000    0.5000         0         0         0         0
    0.2500    1.0000    0.7500         0         0         0
         0    0.5000    1.5000    1.0000         0         0
         0         0    0.7500    2.0000    1.2500         0
         0         0         0    1.0000    2.5000    1.5000
         0         0         0         0    1.2500    3.0000
```

```
         0           0           0           0           0    1.5000

>> Ht*y(:)

ans =

    0.2500
    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
    4.2500
    1.5000

>> Ht * ( y(:) - Hx )

ans =

   -0.2500
   -0.7500
   -1.0000
   -1.0000
   -1.0000
   -1.0000
   -0.7500
   -0.2500

>>
```

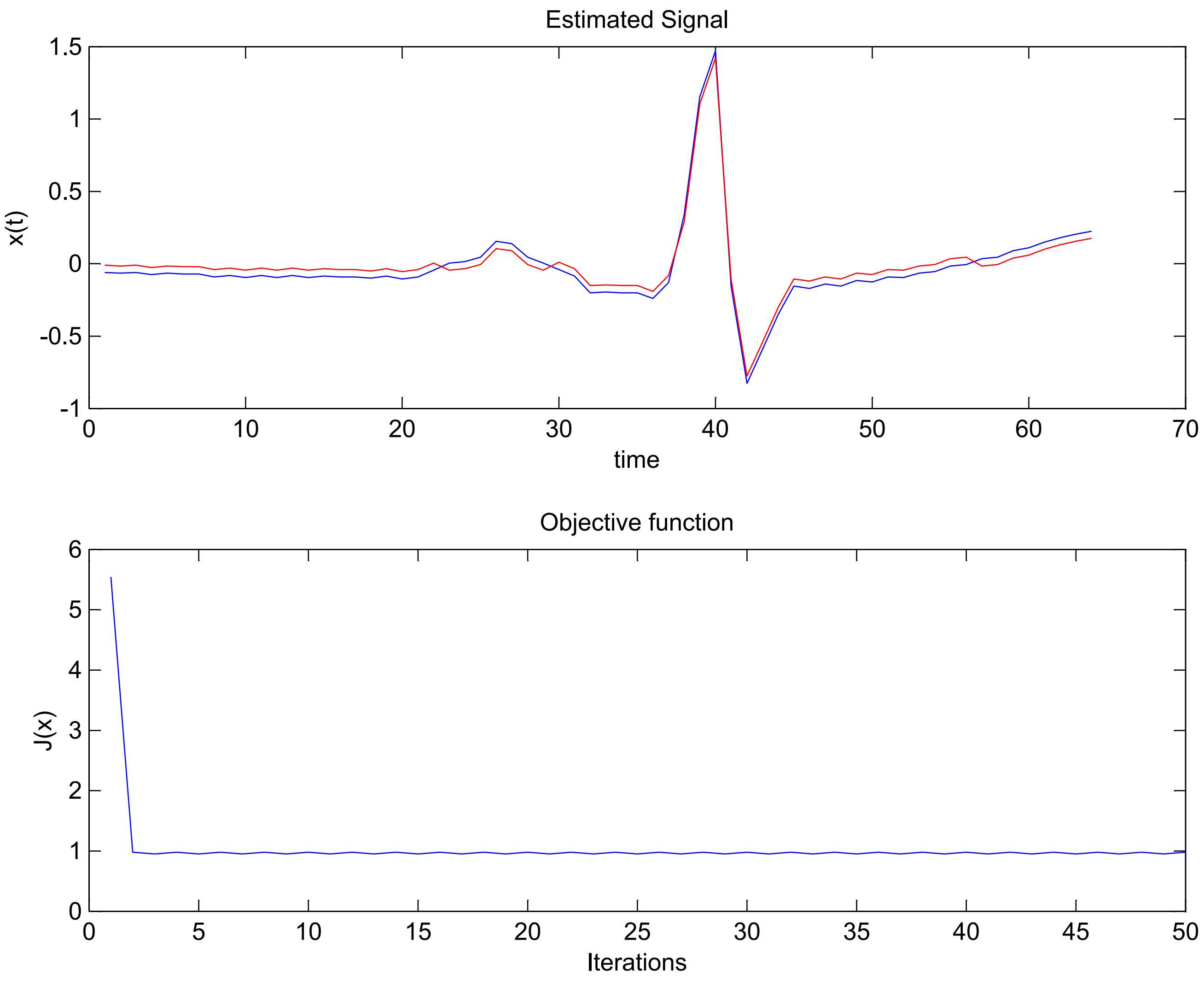
```
>> % now creating a help function as before (ista_ecg.m)
>> function [x, J ] = ista_ecg( y, H, lambda, alpha, Nit )

% [x, J] = ista(y, H, lambda, alpha, Nit)
% L1-regularized signal restoration using the iterated
% soft-thresholding algorithm (ISTA)
% Minimizes J(x) = norm2(y-H*x)^2 + lambda*norm1(x)
% INPUT
% y - observed signal
% H - matrix or operator
% lambda - regularization parameter
% alpha - need alpha >= max(eig(H'*H))
% Nit - number of iterations
% OUTPUT
% x - result of deconvolution
% J - objective functionm

J = zeros(1, Nit); % Objective function
x = 0*H'*y(:); % Initialize x
T = lambda/(2*alpha);

for k = 1: Nit
    Hx = H*x;
    J(k) = sum(abs(Hx(:)-y(:)).^2) + lambda*sum(abs(x(:)));
    x = soft(x(:) + (H'*(y(:) - Hx))./alpha, T);
end
end

function [x] = soft(y, T)
x = y - T*sign(y);
end
```

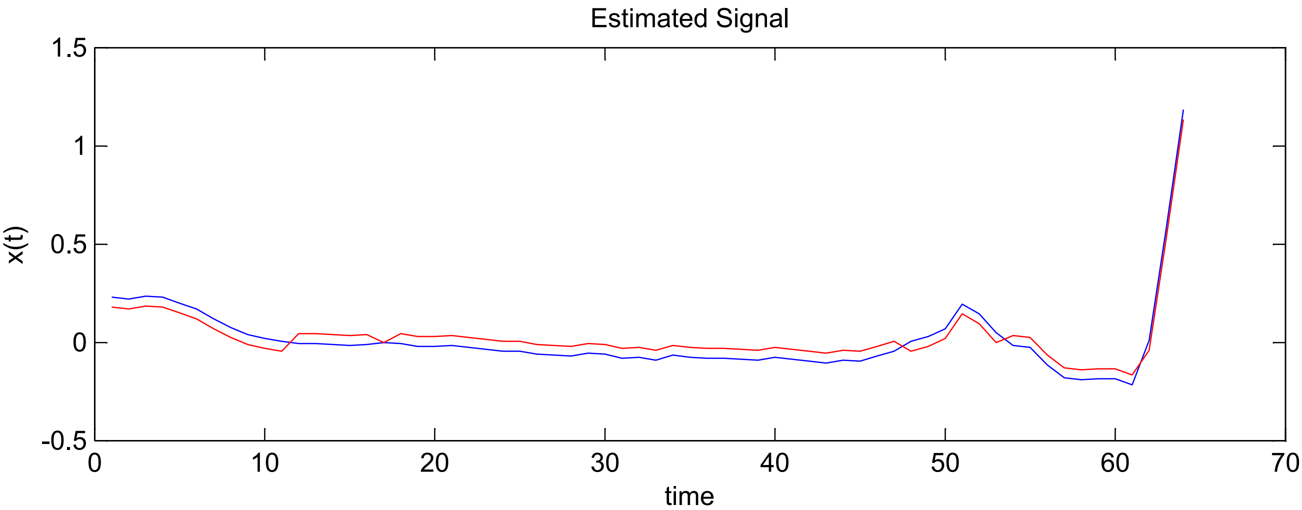
```
>> % using following script to test ISTA (ista_ecg_test,m)
>> clc; clear all; close all;

%-----------Load-Recorded-ECG-----------------------------------------------%
data_a01 = load('a01m.mat');
figure(); Gain = 200;
plotATM('a01m.mat', 'a01m.info');
ecg_data = data_a01.val/Gain; % observed signal
%----------------------------------------------------------------------------%


%-----------Define-Parameters-----------------------------------------------%
fs = 100; Ts = 1/fs; % data sample rate and duration
L = 2^5; % gaussian shape length in samples
Ny = 2^6; % rows of convolution matrix, y - length
Nx = Ny + L - 1; % rows of convolution matrix, x - length
M = fix(length(ecg_data)/Nx); % number of windows
%----------------------------------------------------------------------------%


%----------------------------------------------------------------------------%
% vector windowing into sections
% y is a matrix with rows of data
[y, padded] = vec2mat(ecg_data, Ny);

% gaussian shape (GS) time window vector
t = -Ts*L/2 : Ts : Ts*(L/2-1); % [sec]

lambda = 0.1; alpha = 1; sig = 0.001; % convergence parameters
Nit = 50; % num of iterations

h = gaussmf(t, [sig 0]); % gaussian impulse response
h = h./sum(h); % normalizing
H = convmtx(h, Ny); % convolution matrix
%----------------------------------------------------------------------------%


%----------------------------------------------------------------------------%
% windows loop M iterations
%F = zeros(1, length(y)); j = 0;

for i = 1:M

    y_in = y(i,:); % pick up corresponding vector

    [x_est, J] = ista_ecg(y_in, H, lambda, alpha, Nit);


    %------------Results-Printout-----------------------------------------%

    % plot observed signal
    figure();
    subplot(211); plot(y_in);
    xlabel('time'); ylabel('y(t)');
    title('Observed Signal');
```

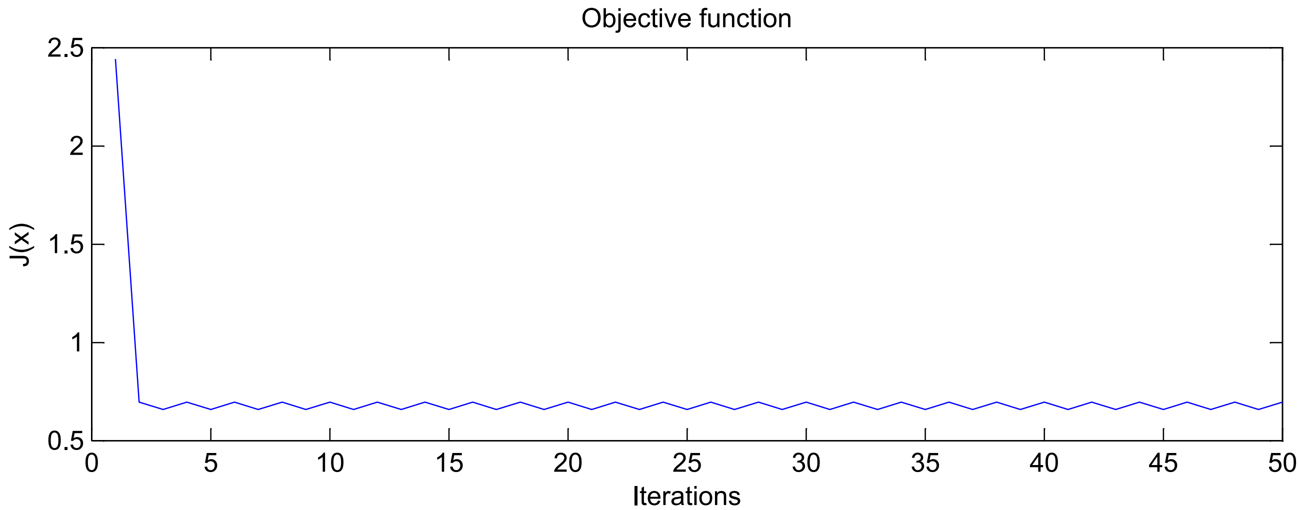
```
    % plot estimated signal
    hold on; plot(x_est(17:80), '--', 'color', 'r');
    xlabel('time'); ylabel('x(t)');
    title('Estimated Signal'); hold off;

    % plot error versus num of iteration
    subplot(212); plot(J);
    xlabel('Iterations'); ylabel('J(x)');
    title('Objective function');
    %-----------------------------------------------------------------------------%

    %F = Film(y_in);
end
%movie(F, 1);
```

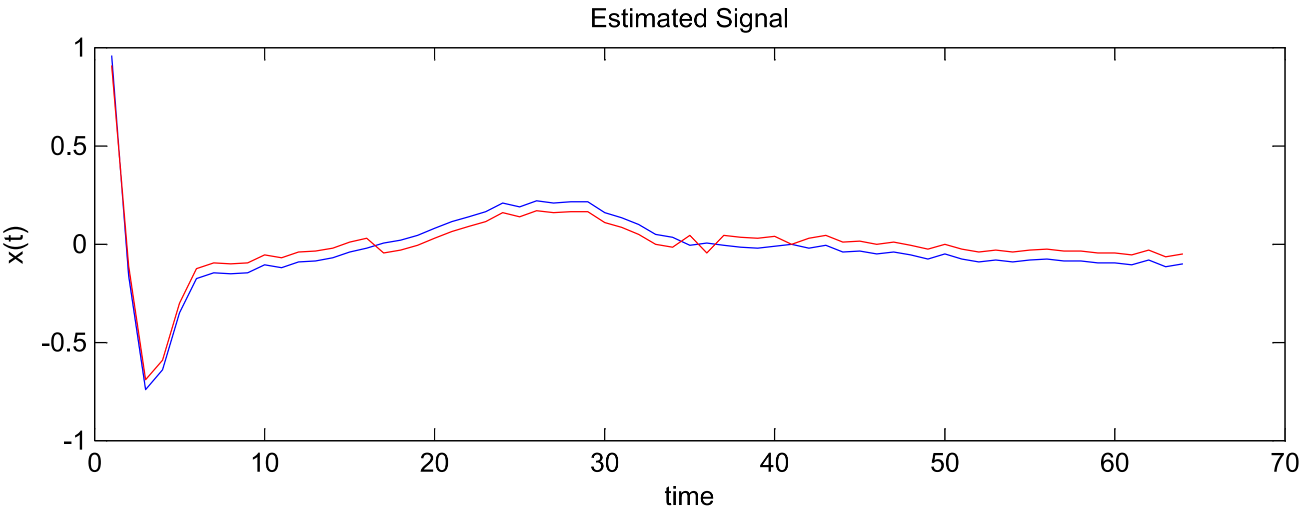


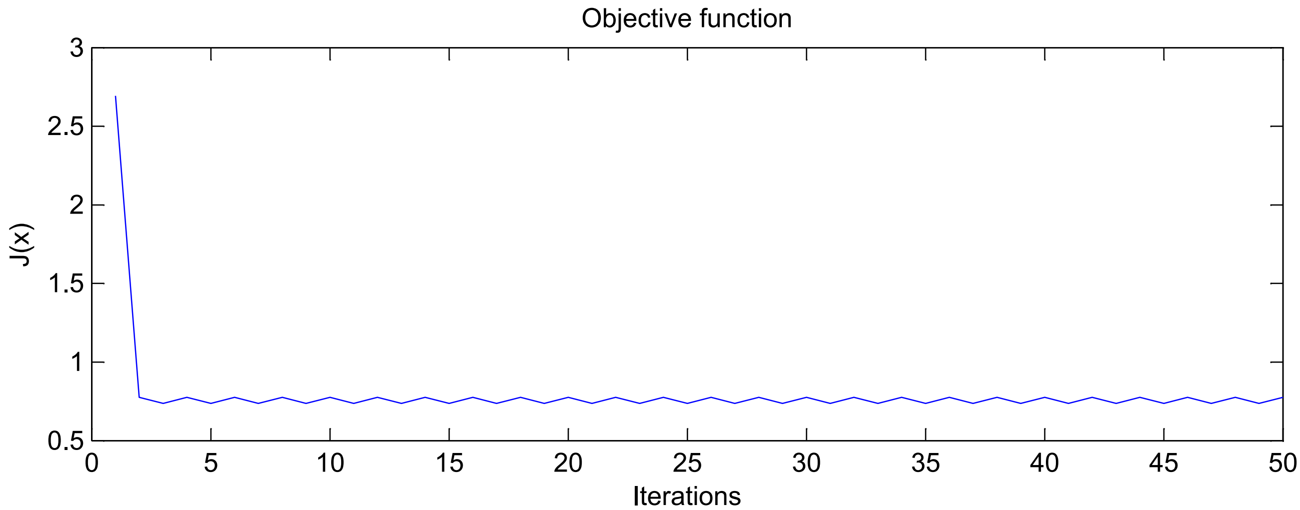
ECG (mV)
2
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10
Time (sec)

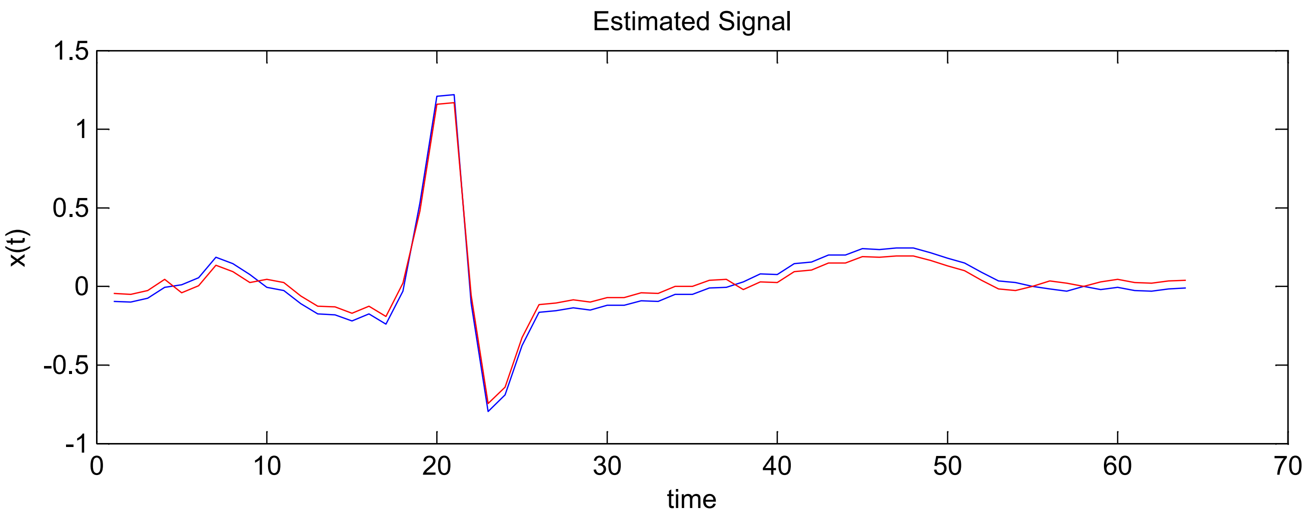

Estimated Signal
x(t)
time
1.5
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70
Objective function
J(x)
Iterations
6
5
4
3
2
1
0
0
5
10
15
20
25
30
35
40
45
50


Estimated Signal
1.5
1
0.5
0
-0.5
x(t)
0
10
20
30
40
50
60
70
time

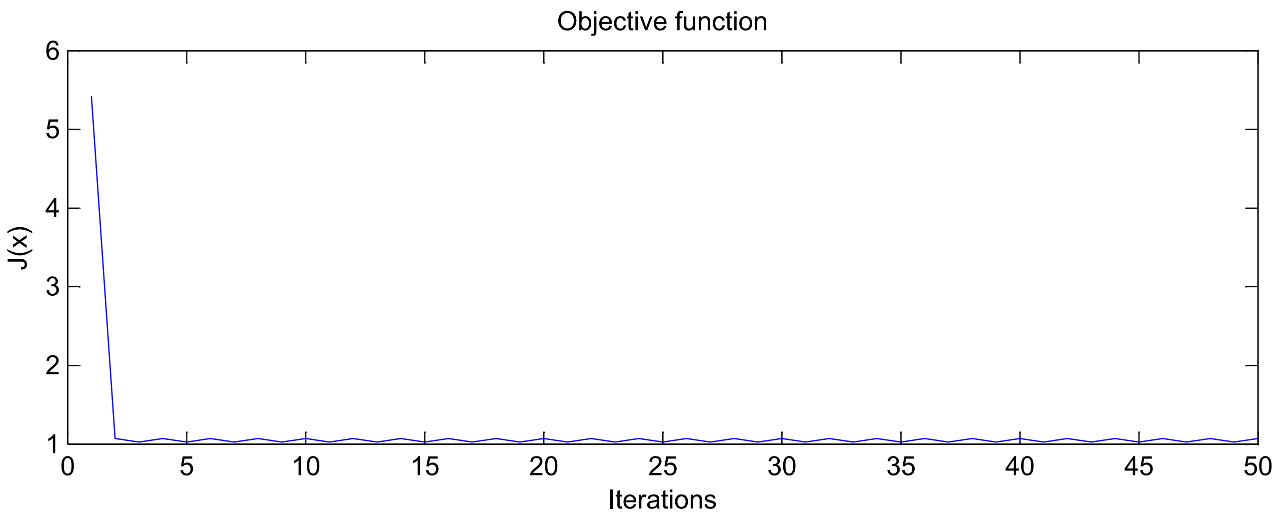

Objective function
2.5
2
1.5
1
0.5
J(x)
0
5
10
15
20
25
30
35
40
45
50
Iterations


Estimated Signal
x(t)
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70
time

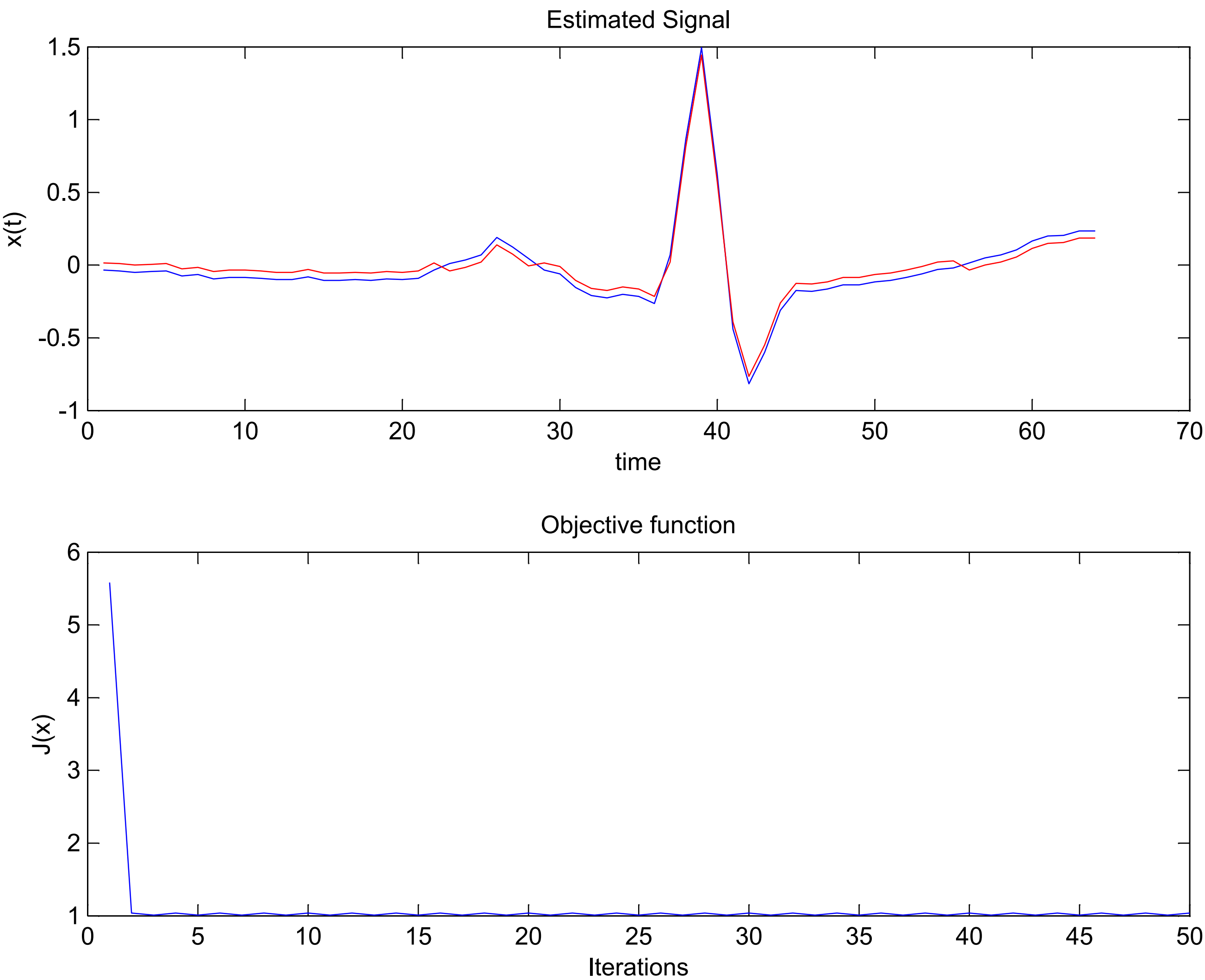

Objective function
J(x)
3
2.5
2
1.5
1
0.5
0
5
10
15
20
25
30
35
40
45
50
Iterations



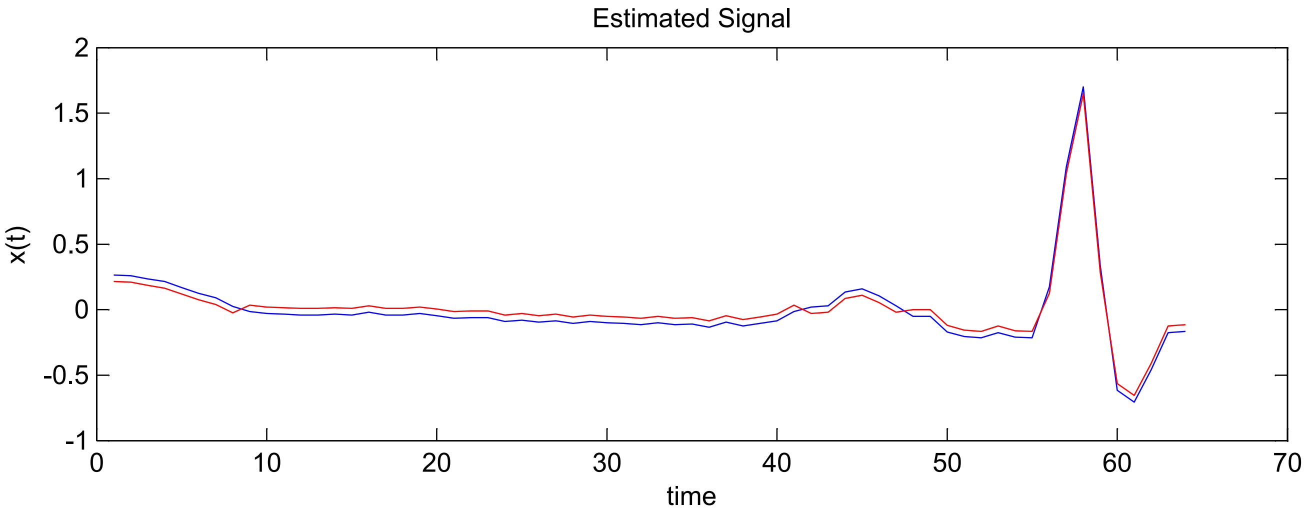
Estimated Signal
x(t)
time
1.5
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70

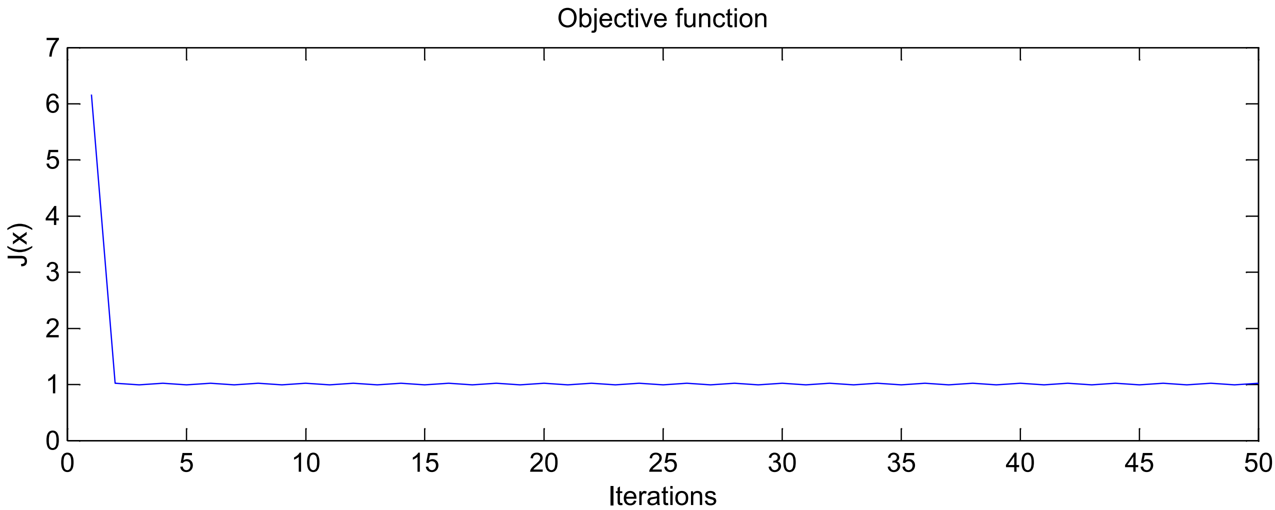


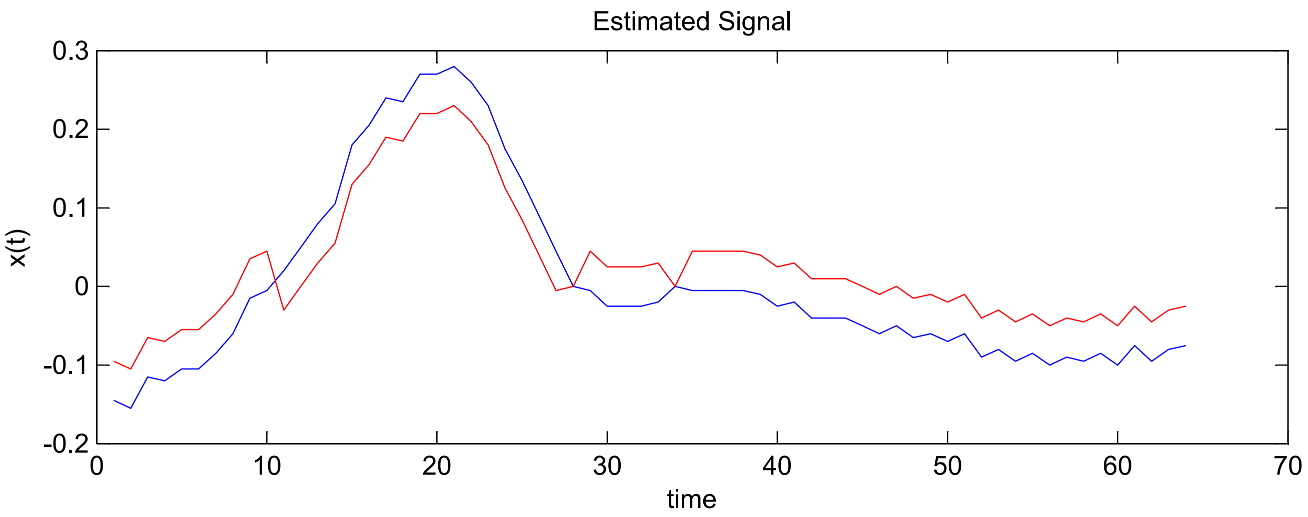
Objective function
J(x)
Iterations
6
5
4
3
2
1
0
5
10
15
20
25
30
35
40
45
50

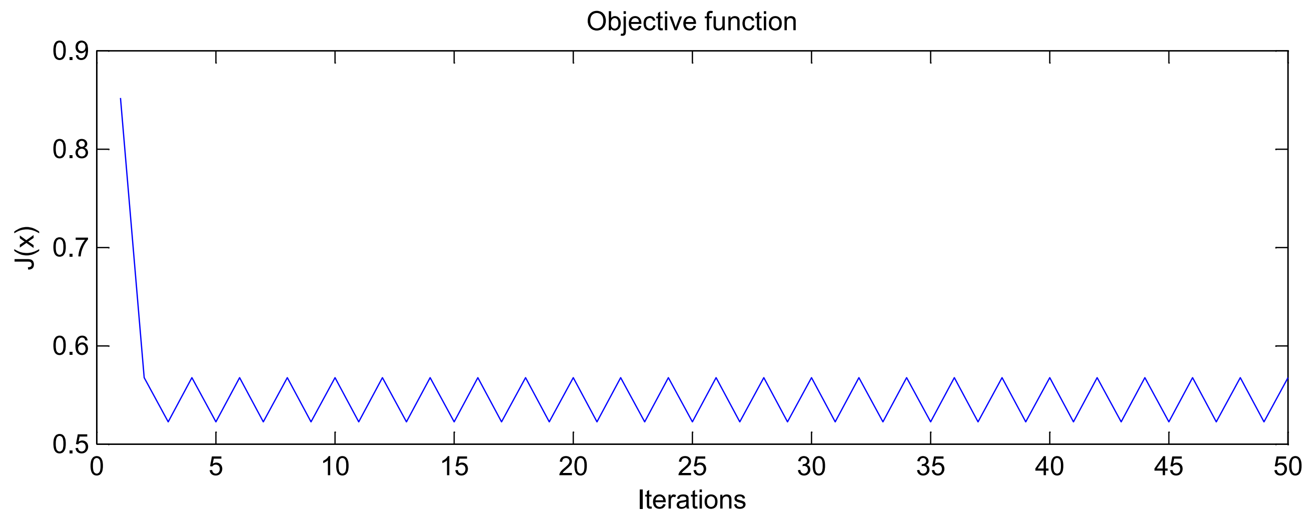


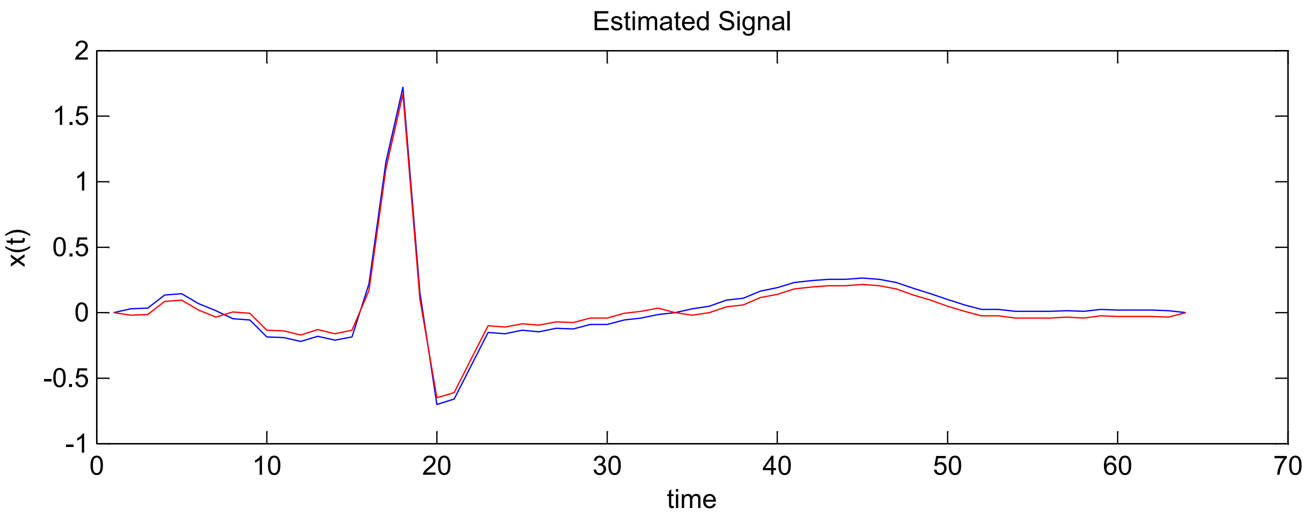
Estimated Signal
x(t)
time
Objective function
J(x)
Iterations


Estimated Signal
x(t)
time
2
1.5
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70

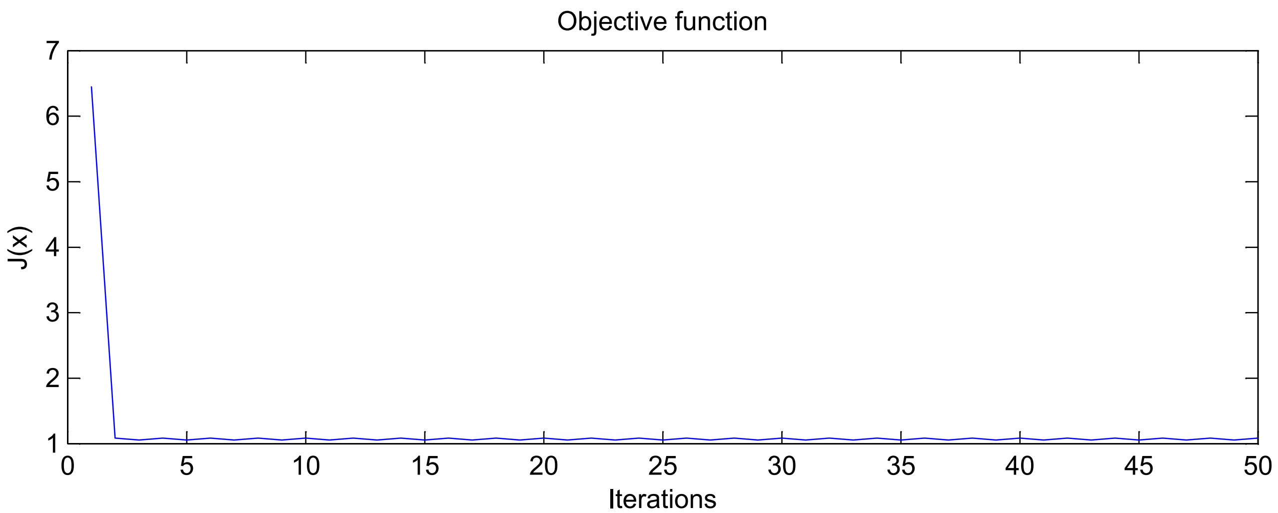

Objective function
J(x)
Iterations
7
6
5
4
3
2
1
0
0
5
10
15
20
25
30
35
40
45
50


Estimated Signal
x(t)
time
0.3
0.2
0.1
0
-0.1
-0.2
0
10
20
30
40
50
60
70

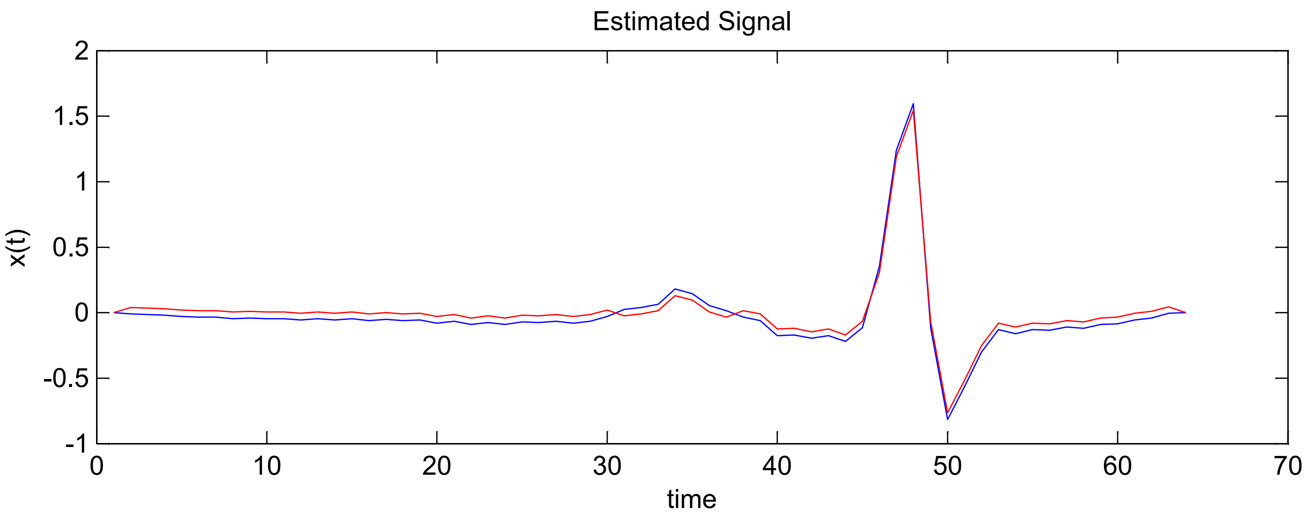

Objective function
J(x)
Iterations
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25
30
35
40
45
50



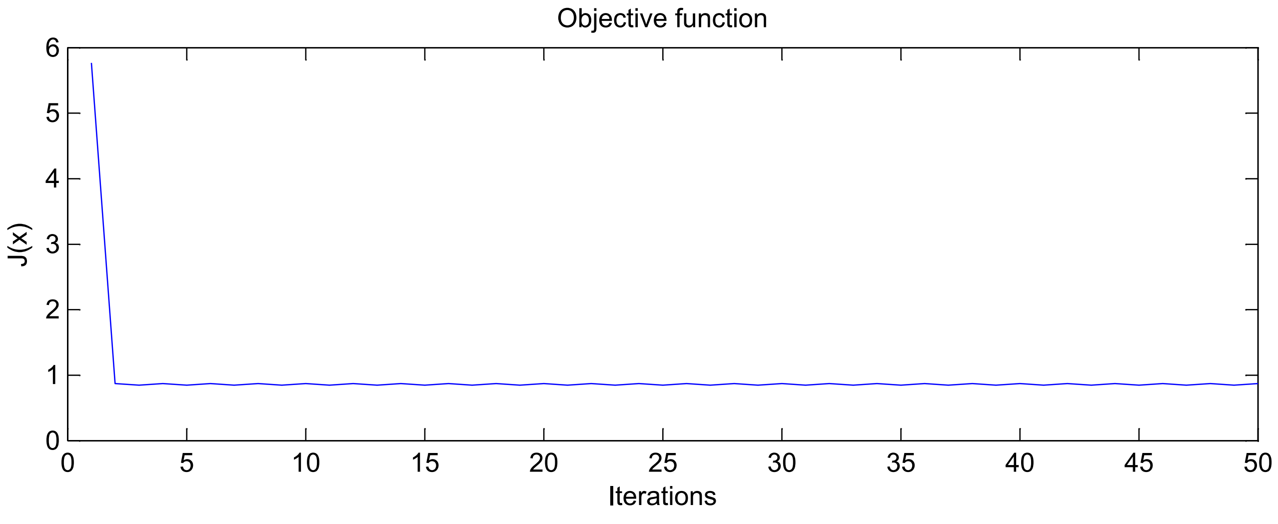
Estimated Signal
x(t)
time
2
1.5
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70



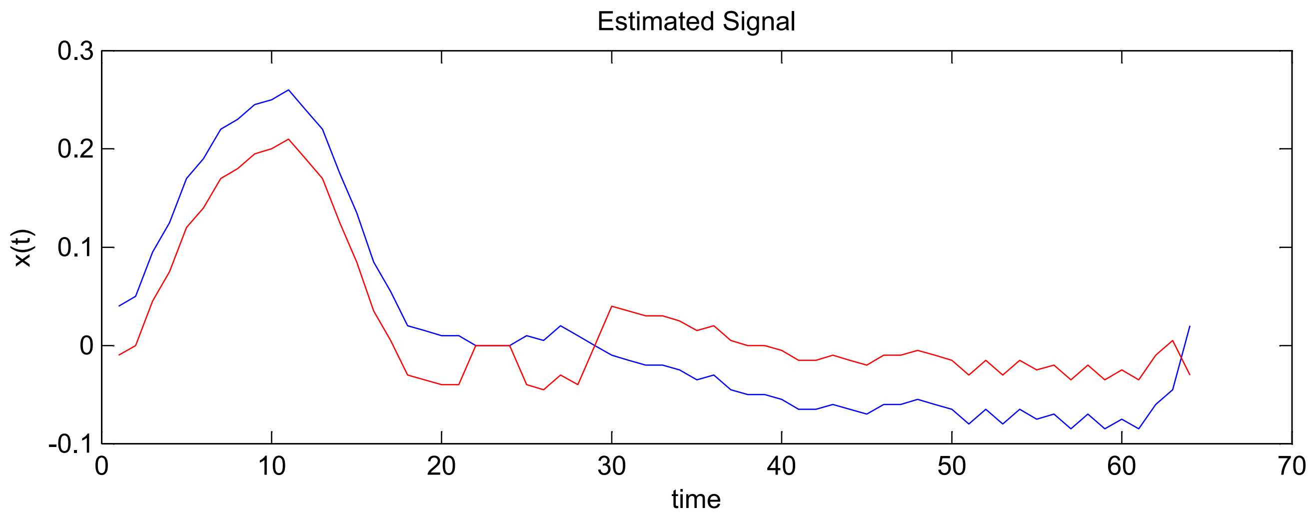
Objective function
J(x)
Iterations
7
6
5
4
3
2
1
0
5
10
15
20
25
30
35
40
45
50

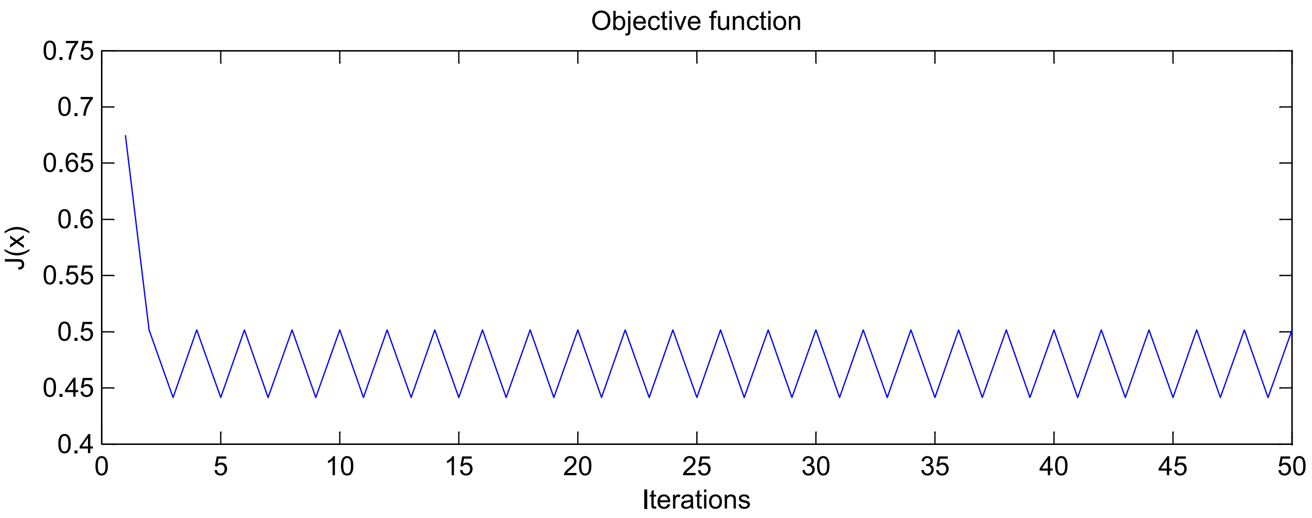


Estimated Signal
x(t)
time



Objective function
J(x)
Iterations


Estimated Signal
x(t)
0.3
0.2
0.1
0
-0.1
0
10
20
30
40
50
60
70
time


Objective function
J(x)
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0
5
10
15
20
25
30
35
40
45
50
Iterations

```
>> % Notes and remarks:
>> % observing final results of ECG signal approximation
>> % we can notice bad approximation results when peaks are smooth
>> % due to gaussian span basis with constant sigma parameter
>> % but on the other hand one can get good results estimating sharp peaks
>> % Possible solution:
>> % find a compromise between amplitudes and sigma parameters for best results
>>
```