The complete set of codes for these solutions are available in the MATLAB m-files `measurement_simulation.m` and `plot_estimates.m`. The relevant codes to each problem are listed below their statements.

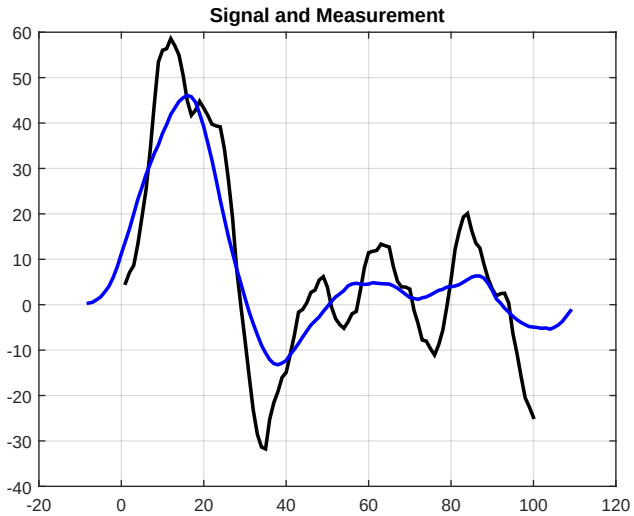
1. Measurement Simulation.

(a) Choose some profile x or generate it randomly.

(b) Create a matrix A.

(c) Simulate a measurement $y = Ax + \nu$.

```
% Problem 1
clear all; %close all;
n = 100;            % Number of sample points of x
n1 = 7;             % Kernel radius of F^(1/2)
n2 = 9;             % Kernel radius for Blurring operator A
s = .1;             % Standard Deviation of measurment noise

% Definitions for various kernels
ker.rectangular = @(d)( ones(1,2*d+1) );
ker.triangular  = @(d)( [1:d+1 d:-1:1] );
ker.parabolic   = @(d)( (d+1)^2-(-d:d).^2 );
ker.gauss       = @(d)( exp(-(-d:d).^2/(2*(d/3)^2)) );

% Kernel for the prior F^(1/2)
b0 = ker.triangular(n1); b0 = b0(((size(b0,2)+1)/2):end); b = [b0 zeros(1,n-size(
    b0,2))];
B = toeplitz(b); % prior F^(1/2)
F = B'*B;        % prior variance

x = B*randn(size(F,1),1);  % Generate x

a = ker.rectangular(n2)/sum(ker.rectangular(n2)); % Kernel for Blurring operator A
A = convmtx(a',n);  % Create convolution matrix

y = A*x + s*randn(size(A,1),1); % Generate noisey data
figure(1), plot(1:n,x,'k-', (1-n2):(n+n2),y,'b-', 'linewidth',2); % plot signal
    and measurement
title('Signal and Measurement');
grid on;
```



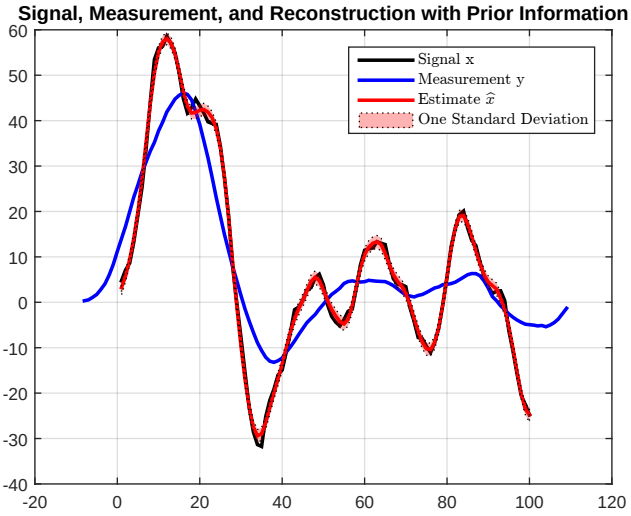
2. Estimation: Construct an optimal linear estimate $\widehat{x}$ and the variance matrix $\mathrm{Var}(\widehat{x}-x)$. Show on the same graph

(a) The original signal x (a curve with components x_i),

(b) Its estimate $\widehat{x}$ (a curve with components $\widehat{x}_i$),

(c) Standard deviations for the estimates $\widehat{x}_i\big(=\sqrt{\mathrm{Var}(\widehat{x}-x)_{ii}}$, can be illustrated by showing the corresponding "corridor" around $\widehat{x}_i\big)$.

```
% Problem 2
Q = inv(1/s*A'*A + inv(F)); % Estimator variance
xhat = Q*(1/s*A'*y); % Estimator
figure(2), plot_estimates(x,n,n2,xhat,Q,y); grid on;
title('Signal, Measurement, and Reconstruction with Prior Information');
```



Note that the reconstruction used prior information F.

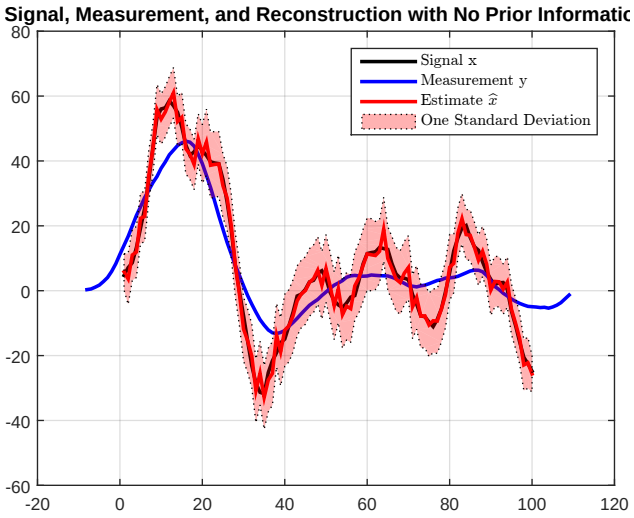
3. Illustrate estimation (Phase 2) in different settings:

(a) Single measurement (y, A, S).

i. Transform (y, A, S) to canonical form (T, v).

ii. Construct the estimate, based on the canonical information.

```
% Problem 3 (a)
get_canonical_data = @(y,A,S)( struct('T',A'*inv(S)*A, ... % Form T matrix
                                      'b',A'*inv(S)*y  ... % Form B'y
                                    ));
info1 = get_canonical_data(y,A,s*eye(n+2*n2));
get_estimates = @(info)( struct('Q',inv(info.T), ...
                                'xhat',(info.T)\info.b ...
                                ));
estimates1 = get_estimates(info1);
figure(3), plot_estimates(x,n,n2,estimates1.xhat,estimates1.Q,y); grid on;
```



Observe that without prior information, there is more uncertainty in the estimate.

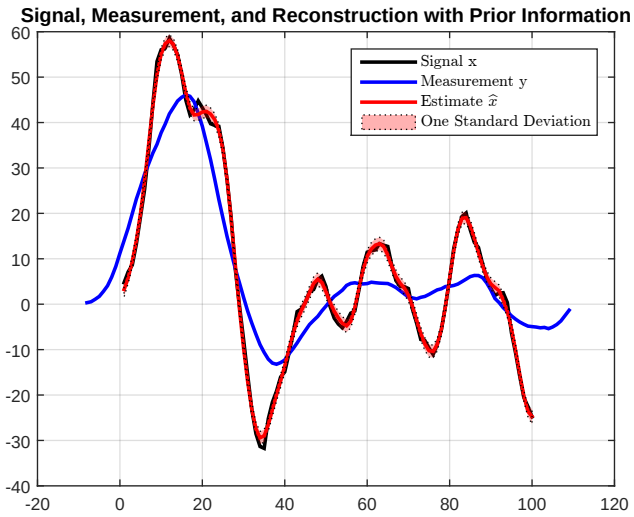
(b) Single measurement (y, A, S) wiht prior information: $x \sim (0, F)$.

i. Transform the measurement and the prior information to canonical form.

ii. Combine pieces of canonical information.

iii. Construct the estimate, based on the combined canonical information.

```
% Problem 3 (b)
combine_information = @(info1,info2)(struct('T',info1.T+info2.T,...
                                            'b',info1.b+info2.b));

prior_to_canonical = @(mu,F)( get_canonical_data(mu,eye(n),F) );
prior_info = prior_to_canonical(zeros(n,1),F);
info2 = combine_information(info1,prior_info);
estimates2 = get_estimates(info2);
figure(4), plot_estimates(x,n,n2,estimates2.xhat,estimates2.Q,y); grid on;
```



Note that this is precicely the same reconstruction in Problem 2.

(c) Many measurements (y_i, A_j, S_j).

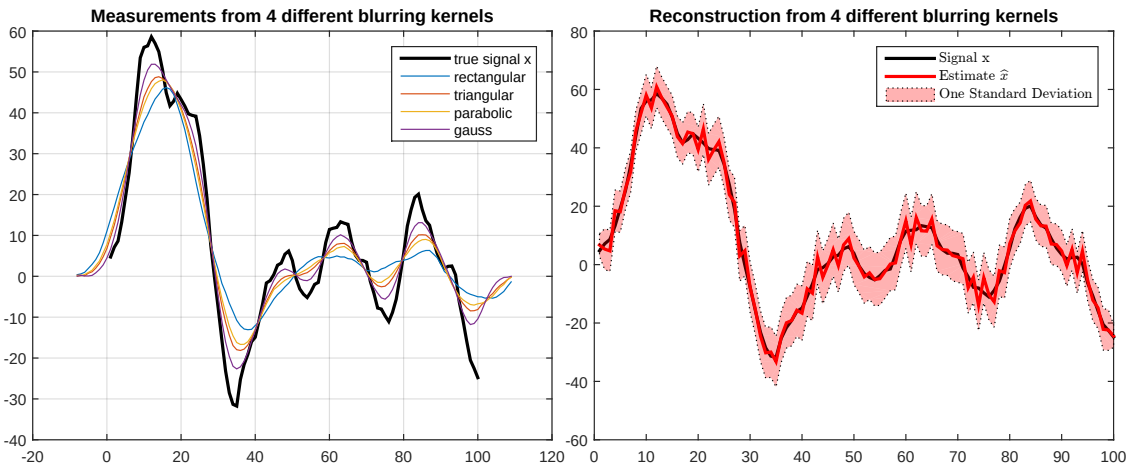
i. Extract canonical information from each measurement.

ii. Combine pieces of canonical information.

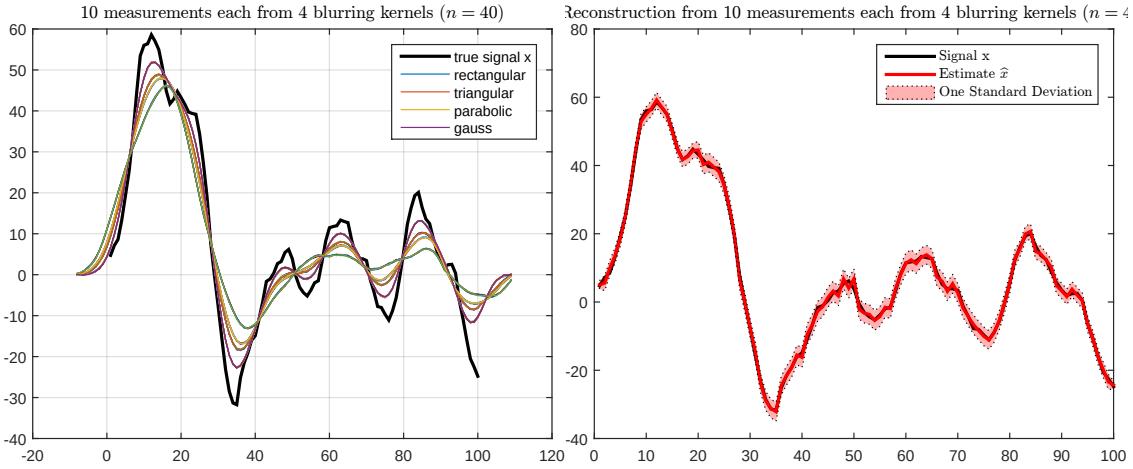
iii. Construct the estimate, based on the combined canonical information.

The following simulates four measurements from each of the four defined blurring kernels in Problem 1. Note that in the implementation, assuming no prior information is the same as starting with an empty canonical representation.

```
% Problem 3 (c)
figure(5);
plot(1:n,x,'k-', 'linewidth',2); % plot signal
hold all;
names = fieldnames(ker);
info = get_canonical_data(zeros(n,1),zeros(n),eye(n)); % No information
for j = 1:length(names);
  a = ker.(names{j})(n2)/sum(ker.(names{j})(n2));
  a = a/sum(a);
  A = convmtx(a',n);  % Create convolution matrix
  y = A*x + s*randn(size(A,1),1); % Generate noisey data
  new_info = get_canonical_data(y,A,s*eye(n+2*n2));
  info = combine_information(info,new_info);
  plot((1-n2):(n+n2),y);
end
hold off;
grid on;
legend(['true signal x';fieldnames(ker)]);
title('Measurements from 4 different blurring kernels');

estimates = get_estimates(info);
figure(6), plot_estimates(x,n,n2,estimates.xhat,estimates.Q);
```



Observe that increasing to four measurements reduces the uncertertainty but not by much. If ten measurements are taken with each of the four "devices" ($n = 40$), we see a substantial decrease in uncertainty, although not as much as including accurate prior information.



(d) Many measurements (y_j, A_j, S_j) with a prior information $x \sim (0, F)$. Same as item (c).

```
% Problem 3 (d)
names = fieldnames(ker);
info = prior_to_canonical(zeros(n,1),F);
for j = 1:length(names);
  a = ker.(names{j})(n2)/sum(ker.(names{j})(n2));
  a = a/sum(a);
  A = convmtx(a',n);  % Create convolution matrix
  y = A*x + s*randn(size(A,1),1); % Generate noisey data
  new_info = get_canonical_data(y,A,s*eye(n+2*n2));
  info = combine_information(info,new_info);
end
estimates = get_estimates(info);
figure(9), plot_estimates(x,n,n2,estimates.xhat,estimates.Q);
```

For this problem, we use the same use the same four simulated measurements from 3c. Note that the only modification to the code is that the loop now starts with the canonical representation of the prior $x \sim (0, F)$.