Real Time Signal Processing with Symmetric and Asymmetric Support Intervals

Kevin Joyce and Lia Harrington
University of Montana

May 14, 2015

Outline

Importance of Real Time Signal Processing

What is real time signal processing?

- Applications
  - Speech recognition
  - Audio signal processing
  - Video compression
  - Weather forecasting
  - Economic forecasting
  - Medical imagining (e.g., CAT, MRI)
  - And more...

Outline

Introduction and Motivation

Problem Outline

Problem Approach and Steps

Code Comments

Results and Conclusions

What is the problem?

Goal: We wish to reconstruct some generated signal $\hat{x}$ that has been distorted by some error and convolution processes.

Solution: Find the “convolution inverse” of a. We call this the reconstruction operator R.

Reconstruction Operator R

- Following Lecture 13, we seek a linear reconstruction operator R that we assume is given by convolution with some r supported on a specified interval Δ, so that $\widehat{x} = r * x$.
- It was shown that such an operator satisfies

$$H(r) = E(\widehat{x} - x)^2 = \left\langle P(r - P^{-1}q), r - P^{-1}q \right\rangle_{\Delta} + f_0 - \left\langle q, P^{-1}q \right\rangle_{\Delta}$$

  where P is the operator associated with convolution by $p = a * \phi * a^* + \sigma^2 \delta$ and $q = a * \phi$.
- So, for a given Δ support for r, the reconstruction kernel is uniquely determined by $r = P^{-1}q|_{\Delta}$, and

$$\mathrm{Var}\,\widehat{x} = H_{min} = f_0 - \left\langle q, P^{-1}q \right\rangle_{\Delta}.$$

Outline

Problem Approach Overview

1. Specify problem settings
   - $y = a * x + \nu, \ \ \nu \ (0, \sigma^2 \delta).$
   - $\hat{x} = r * y$
     - $r = P^{-1} q$
   - Choose some $\Delta = [-d, d]$ $\rightarrow$ compute $r = P^{-1} q|_\Delta$
   - Choose some $\Delta = [T, \tau]$ $\rightarrow$ compute $r = P^{-1} q|_\Delta$
2. Compute "optimal" Δ for r by
   - $H(\Delta) = E(\hat{x}_i - x_i)^2 = f_0 - \langle q, P^{-1} q \rangle_\Delta$
   - Plot $H(\Delta)$ vs either d in the symmetric case, or (T, τ) in the asymmetric case.
3. Choose "optimal" Δ $\rightarrow$ compute $\widehat{x} = r * y$.
4. Illustrate the result of estimation with uncertainty given by
   - $\sqrt{E(\hat{x}_i - x_i})^2 = \sqrt{H}$

Problem Strategy: Step 1

Specify the main ingredients of simulated measurement system:

- Specify point spread function (influence function) a,
  - Symmetric
  - Asymmetric
- Covariance function ϕ for the signal x:

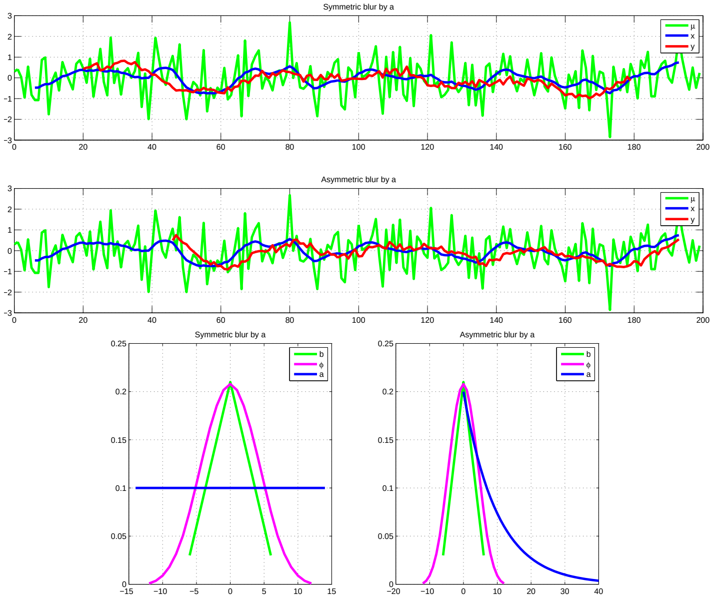
$$\phi = \mathrm{Cov}\,(x) = b * b^* \tag{1}$$

- Variance σ^2 of the independent components of the additive random noise ν

$$y = a * x + \nu, \;\; \nu \; (0, \sigma^2 \delta). \tag{2}$$

- Choose support interval $\Delta = [-d, d]$ or $\Delta = [-T, \tau]$

Signal and Covariance Setup



Problem Simulation

Specify the main ingredients of simulated measurement system:

- Finitely supported point spread function (influence function),
  - Symmetric case: $a_i = \frac{1}{10}$ for $|i| < 15$.
  - Asymmetric case: $a_i = \frac{2}{10}e^{-i/40}$ for $0 \leq i \leq 40$.
- The covariance function ϕ for the signal x is given by

$$\phi = \mathrm{Cov}\,(x) = b * b^*$$

  where $b_i = \frac{21}{100}(1 - |i|)$ for $|i| \leq 7$.
- Measurement noise is modeled with a zero mean Gaussian ν with a specified $\sigma^2 = \frac{1}{100}$.
- Finally, the data is given by

$$y = a * x + \nu$$

Outline

Finitely Supported Function Data Type

- The fundamental data type in `Matlab` is the “matvec” whose operations are not “natural” for finitely supported discrete functions.
- `Matlab` supports object oriented programming which allows for the creation of a custom data type for which we can implement convolution as the natural multiplication.

```
classdef FinSupFun % Finite Support Function
    properties (SetAccess = private)
...
    end

    methods
...
    end
end
```

Finitely Supported Function Data Type

- From the demo codes provided, convolution and addition were "overrided" to allow statements like

```
mu = FinSupFun(randn(1,N),0); % Construct finitely
    supported white noise
x = b .* mu; % Inner convolution
y0 = a .* x;
y = y0 + FinSupFun(s*randn(size(y0.f)),y0.l);
phi = b*b'; % Outer convolution
p = a*phi*a' + FinSupFun(s^2);
q = phi*a';
```

- Note that the $*$ and $.*$ operations now represent convolution between `FinSupFun` objects.
- This makes code easier to read and debug.

Finitely Supported Function Data Type

- We added the following methods

```
function c = restricted_to(a,l,r) % Restrict support to [l,r], if [l,r] is
      bigger than [a.l,a.r], then pad with zeroes.
    L = max(l,a.l); % Left endpoint of restricted interval
    R = min(r,a.r); % Right endpoint of restricted interval
    ...
end

function c = mldivide(a,b) % \ De-convolution by constructing toeplitz matrix. a
      must be symmetric
  n = length(b.f);
  toeplitz_row = [a.f((a.r+1):end), zeros(1, n-(a.r))];% This needs to be from
       the center of p and padded with zeros
  ...
end
```

- so computing $P^{-1}q$ on a restriced interval are easily implemented as follows

```
q_delta1 = q.restricted_to(-d,d); % Restrict (or zero pad) to [-d,d].
q_delta2 = q.restricted_to(-T,tau); % Restrict (or zero pad) to [-T,tau].
r1 = p \ q_delta1;
r2 = p \ q_delta2;
```

Outline

References

[1] Golubtsov, P. (2015). Theoretical Big Data Analytics course notes.