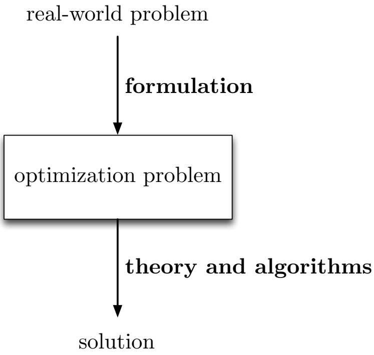
Optimization and algorithms

Part 1: the art of formulating optimization problems

João Xavier
email: jxavier@isr.ist.utl.pt

Instituto Superior Técnico, Universidade de Lisboa


real-world problem
formulation
optimization problem
theory and algorithms
solution

An optimization problem is a mathematical object of the following form:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & h_1(x) = 0 \\ & \vdots \\ & h_p(x) = 0 \\ & g_1(x) \leq 0 \\ & \vdots \\ & g_m(x) \leq 0 \end{array}$$

- $x \in \mathbf{R}^n$ is the optimization variable
- $f : \mathbf{R}^n \to \mathbf{R}$ is the objective or cost function that we want to minimize
- $h_1, \ldots, h_p, g_1, \ldots, g_m : \mathbf{R}^n \to \mathbf{R}$ are constraint functions

Outline

Examples of problem formulations:

- choosing a portfolio
- scheduling aircraft landings
- packing a suitcase
- controlling two robots
- denoising a piecewise constant information signal

Example: choosing a portfolio

- you have T euros to invest
- you can invest in n stocks
- r_i is the *expected* rate of return for stock $i = 1, \ldots, n$
- x_i is the amount you invest in stock $i = 1, \ldots, n$
- for a given investment $x = (x_1, x_2, \ldots, x_n)$, you are *expected* to receive

$$r_1 x_1 + r_2 x_2 + \cdots + r_n x_n$$

- how do you choose the best portfolio x?

- problem formulation:

$$\begin{array}{ll} \underset{x \in \mathbf{R}^n}{\text{maximize}} & r_1 x_1 + r_2 x_2 + \cdots + r_n x_n \\ \text{subject to} & x_1 + \cdots + x_n = T \\ & x_i \geq 0, \quad i = 1, \ldots, n \end{array}$$

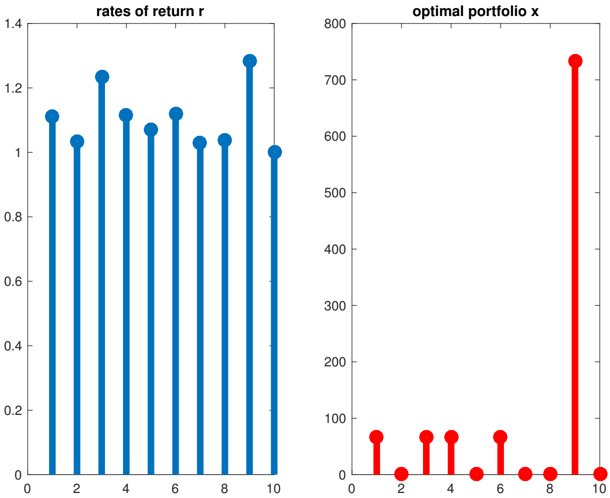
- optimal solution places all the money on the most attractive stock
- let's introduce a diversity constraint: no more than 80% of the investment should be concentrated in any two stocks

- problem formulation:

$$
\begin{array}{ll}
\underset{x \in \mathbf{R}^n}{\text{maximize}} & r_1 x_1 + r_2 x_2 + \cdots + r_n x_n \\
\text{subject to} & x_1 + \cdots + x_n = T \\
& x_i \geq 0, \quad i = 1, \ldots, n \\
& x_i + x_j \leq 0.8T, \quad i \neq j
\end{array}
$$

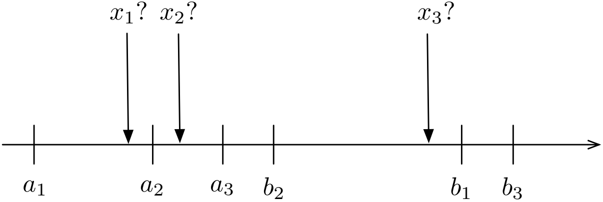
```
% portfolio.m; uses package CVX from http://cvxr.com/cvx
n = 10; % number of stocks
r = 1+0.3*rand(n-1,1); % generate random returns
r = [ r ; 1 ]; % the last one is a risk-free asset
T = 1000; % set budget

% solve the optimization problem
cvx_begin quiet
   variable x(n);
   maximize(r'*x);

   %subject to
   x ≥ 0; sum(x) == T;
   for i = 1:n
     for j = i+1:n
        x(i) + x(j) ≤ 0.8*T;
     end;
   end;
cvx_end;

figure(1); clf; % plot solution
subplot(1,2,1); stem(r,'LineWidth',5);
title('rates of return r');
subplot(1,2,2); stem(x,'r','LineWidth',5);
title('optimal portfolio x');
%print -depsc portfolioexample;
```

- a random example:



Example: scheduling aircraft landings

- n airplanes must land in the order $1 \to 2 \to \cdots \to n$
- airplane i must land in the time interval $[a_i, b_i]$
- x_i is time of landing for airplane i
- how should we choose the landing times?



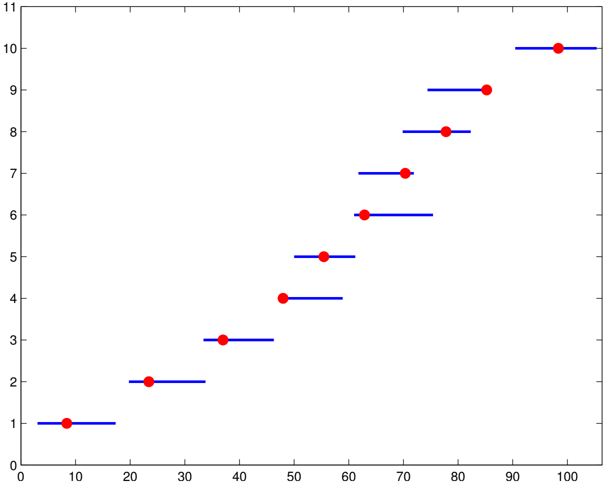
- optimization problem (maximize safety margin):

$$\begin{array}{ll} \underset{x}{\text{maximize}} & \underbrace{\min\{x_2 - x_1, x_3 - x_2, \ldots, x_n - x_{n-1}\}}_{f(x_1,\ldots,x_n)} \\ \text{subject to} & x_i \le x_{i+1}, \quad i = 1, \ldots, n-1 \\ & a_i \le x_i \le b_i, \quad i = 1, \ldots, n \end{array}$$

- optimization variable is $x = (x_1, x_2, \ldots, x_n) \in \mathbf{R}^n$ (landing times)

```
% uses package CVX from http://cvxr.com/cvx
n = 10; % choose n = number of planes
a = sort(100*rand(n,1)); % generate landing intervals
b = a+10+5*rand(n,1);

figure(1); clf; % plot landing intervals
for i=1:n
  plot([a(i) b(i)],[ i i],'LineWidth',2); hold on;
end;
axis([ 0 max(b)+1 0 n+1 ]);

% solve optimization problem
cvx_begin quiet
   variable x(n,1);

   % build cost function
   f= x(2)-x(1);
   for i = 3:n
       f= min(f,x(i)-x(i-1));
   end;
   maximize(f);
   % subject t
   x(1:n-1) ≤ x(2:n); x ≥ a; x ≤ b;
cvx_end;
plot(x,1:n,'r.','MarkerSize',25); % plot solution
```

- typical output:



Example: packing a suitcase

- you will travel with a suitcase that has volume V
- by airline company regulations, your filled suitcase cannot weight more than W
- you would want to carry n items
- item i $(i = 1, \ldots, n)$ costs c_i, has volume v_i, and weighs w_i
- which items do you choose to put in the suitcase?

- use binary variables to encode decision: $x_i = 1$ means item i goes into the suitcase; $x_i = 0$ means item i does not go into the suitcase

- problem formulation:

$$\begin{array}{ll} \underset{x \in \mathbf{R}^n}{\text{maximize}} & \sum_{i=1}^n c_i x_i \\ \text{subject to} & \sum_{i=1}^n x_i v_i \leq V \\ & \sum_{i=1}^n x_i w_i \leq W \\ & x_i \in \{0,1\}, \quad i = 1, \ldots, n \end{array}$$

```
% suitcase.m; uses Matlab optimization toolbox

% choose n = number of items
n = 20;

% generate random volumes, weights, and costs
v = 20*rand(n,1);
w = 3*rand(n,1);
c = 1000*rand(n,1);

% can only carry 70% (80%) of total volume (weigth) of items
maxV = 0.7*sum(v);
maxW = 0.8*sum(w);

Aineq = [  v' ; w' ; -eye(n) ; eye(n) ];
bineq = [ maxV ; maxW ; zeros(n,1) ; ones(n,1) ];
% solve the optimization problem
x = intlinprog(-c,1:n,Aineq,bineq)',

figure(1); clf;
stem(v,'LineWidth',5); title('volumes'); hold on;
figure(2); clf;
stem(w,'LineWidth',5);
figure(3); clf;
stem(c,'LineWidth',5); title('costs');
```

Example: controlling two robots

- we are going to control two robots between time $\tau = 0$ and time $\tau = \tau_{\mathsf{f}}$
- at $\tau = 0$, robot i is resting at position $s_i \in \mathbf{R}^2$
- at $\tau = \tau_{\mathsf{f}}$, robot i should rest at position $t_i \in \mathbf{R}^2$
- way-point constraint: at $\tau = \tau_i$, robot i should rest at position $r_i \in \mathbf{R}^2$
- wireless constraint: at all times $0 \leq \tau \leq \tau_{\mathsf{f}}$, the distance between the robots should be less or equal to d
- to move the robots, we apply forces
- "energy" of force $f_i(\tau)$ is $\|f_i(\tau)\|^2$
- which forces meet all the constraints and have the least energy?

- m_i is mass of robot i
- $p_i(\tau) \in \mathbf{R}^2$ is position of robot i at time τ
- $v_i(\tau) \in \mathbf{R}^2$ is velocity of robot i at time τ
- $f_i(\tau) \in \mathbf{R}^2$ is force that we apply to robot i at time τ
- how does robot i behave when we apply a force?
- Newton's law says

$$\begin{aligned} m_i \frac{dv_i(\tau)}{d\tau} &= f_i(\tau) - \beta v_i(\tau) \\ \frac{dp_i(\tau)}{d\tau} &= v_i(\tau) \end{aligned}$$

- $\beta > 0$ is drag coefficient

- sampling each h secs. gives (approximate) discrete-time model:

$$
\begin{aligned}
m_i \frac{v_i((t+1)h) - v_i(th)}{h} &= f_i(th) - \beta v_i(th) \\
\frac{p_i((t+1)h) - p_i(th)}{h} &= v_i(th)
\end{aligned}
$$

- in vector notation:

$$
\underbrace{\begin{bmatrix} p_i((t+1)h) \\ v_i((t+1)h) \end{bmatrix}}_{x_i(t+1)} = \underbrace{\begin{bmatrix} I_2 & hI_2 \\ 0 & \left(1 - \frac{\beta h}{m_i}\right) I_2 \end{bmatrix}}_{A_i} \underbrace{\begin{bmatrix} p_i(th) \\ v_i(th) \end{bmatrix}}_{x_i(t)} + \underbrace{\begin{bmatrix} 0 \\ \frac{h}{m_i} I_2 \end{bmatrix}}_{B_i} \underbrace{f_i(th)}_{u_i(t)}
$$

- $x_i(t) \in \mathbf{R}^4$ is state of robot i at discrete-time $t = 0, 1, 2, \ldots$

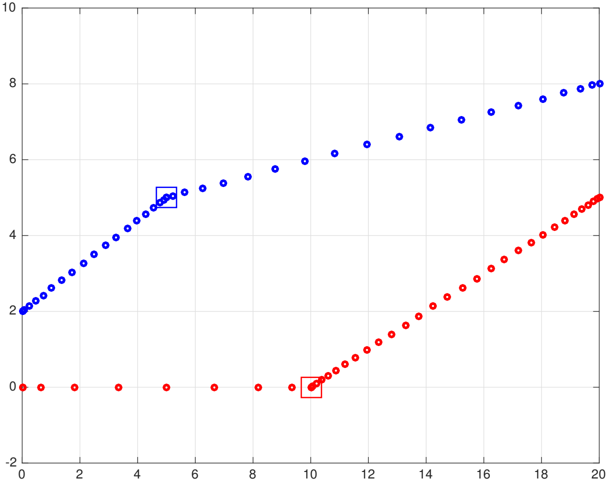
- assume $\tau_i = hT_i$ and $\tau_\mathsf{f} = hT_\mathsf{f}$ for some integers T_1, T_2 and T_f

- problem formulation:

$$
\begin{array}{ll}
\underset{x_i(t),u_i(t)}{\text{minimize}} & \sum_{i=1}^{2} \sum_{t=0}^{T_\mathsf{f}-1} \|u_i(t)\|^2 \\
\text{subject to} & x_i(t+1) = A_i x_i(t) + B_i u_i(t), \quad t = 0, 1, \ldots, T_{\mathsf{f}-1} \\
& x_i(0) = (s_i, 0)\,, \quad i = 1, 2 \\
& x_i(T_\mathsf{f}) = (t_i, 0)\,, \quad i = 1, 2 \\
& x_i(T_i) = (r_i, 0)\,, \quad i = 1, 2 \\
& \left\| \begin{bmatrix} I_2 & 0 \end{bmatrix} (x_1(t) - x_2(t)) \right\| \le d, \quad t = 0, 1, \ldots, T_\mathsf{f}
\end{array}
$$

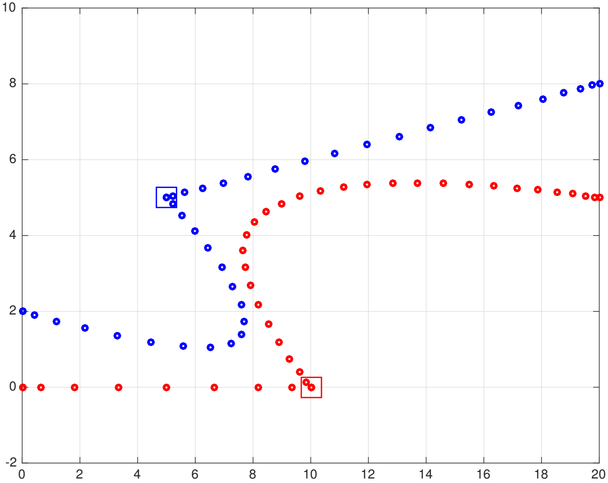
- the optimization variables are $x_i(t), u_i(t)$ for $i = 1, 2$ and $t = 0, 1, \ldots, T_\mathsf{f}$

- example without the distance constraint:



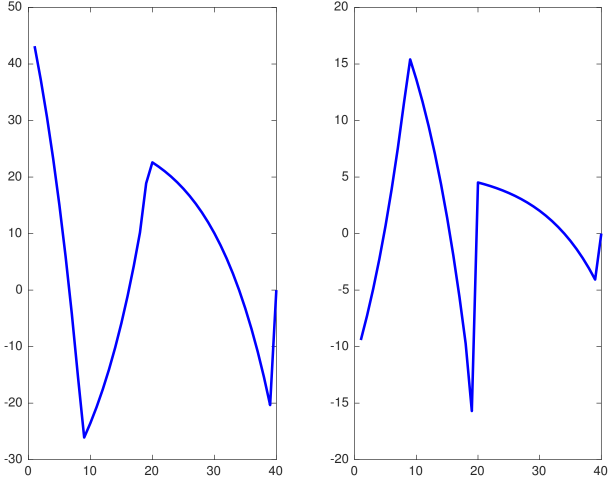
Optimal robots' positions $p_1(t)$ and $p_2(t)$ for $t = 0, 1, \ldots, T_{\mathsf{f}}$

- same example with the distance constraint:

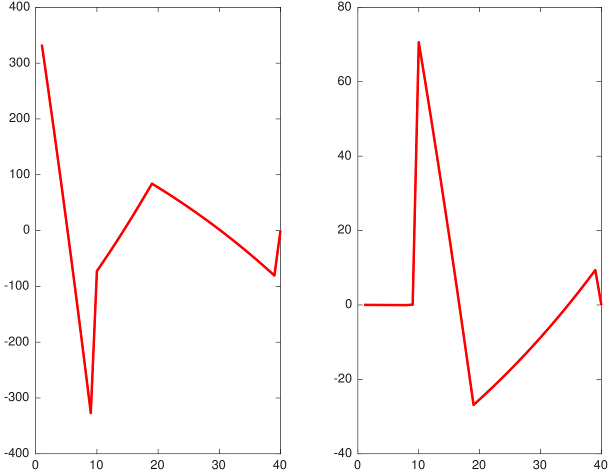


Optimal robots' positions $p_1(t)$ and $p_2(t)$ for $t = 0, 1, \ldots, T_{\mathsf{f}}$

- optimal control sequence for robot 1: $u_1(t) \in \mathbf{R}^2$ for $t = 0, 1, \ldots, T_{\mathsf{f}}$



- optimal control sequence for robot 2: $u_2(t) \in \mathbf{R}^2$ for $t = 0, 1, \ldots, T_\mathsf{f}$



Denoising a piecewise constant signal

Consider an information signal contaminated with additive noise:

$$\underbrace{\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(T) \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(T) \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} v(1) \\ v(2) \\ \vdots \\ v(T) \end{bmatrix}}_{v}$$

where

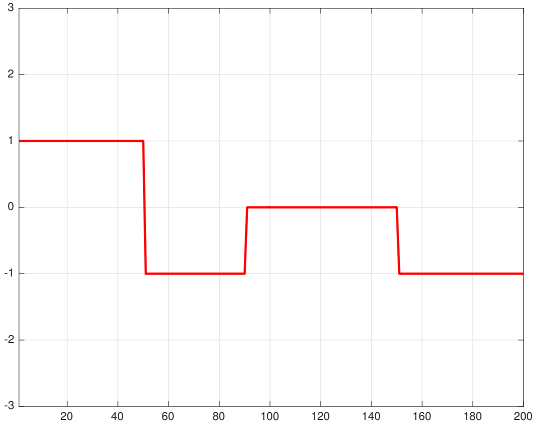
- y is the observed signal (known)
- x is the information signal (unknown)
- v is the additive noise signal (unknown)

Given that you know signal y, how do you guess the unknown signal x?

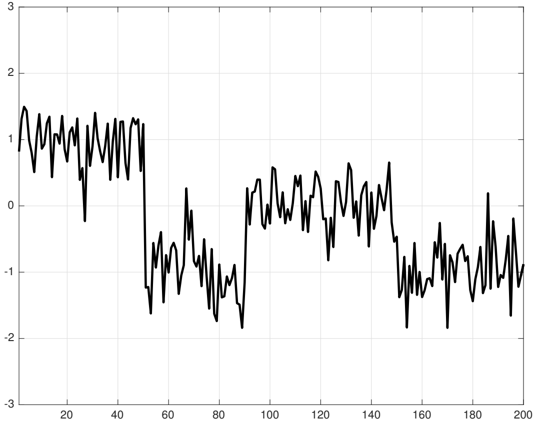
Assume

- the noise signal v is a "small" vector that fluctuates about zero
- the information signal x is piecewise constant (but with unknown switching times and amplitudes)

Example of an information signal x:

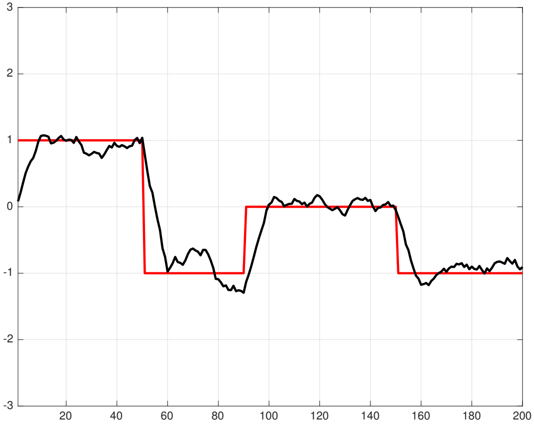


Corresponding example of an observed signal y :



Low-pass filtering approach:

$$\widehat{x}(t) = \frac{1}{10}\left(y(t) + y(t-1) + \cdots + y(t-9)\right)$$



The low-pass estimate blurs the sharp transitions

Key-point: x being piecewise constant means

$$x(t) - x(t-1) = 0$$

for most of the ts

Optimization-based approach:

$$\begin{array}{ll} \underset{x,v}{\text{minimize}} & \underbrace{\underbrace{\frac{1}{2}\|v\|^2}_{\text{make } v \simeq 0} + \lambda \underbrace{\sum_{t=2}^{T} |x(t) - x(t-1)|^p}_{\text{make } x[t]-x[t-1]=0 \text{ most of the } t\text{s}}}_{f(x,v)} \\ \text{subject to} & y = x + v \end{array}$$

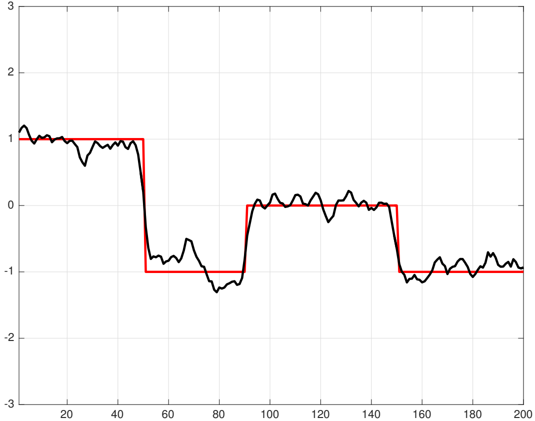
- the optimization variables are x and v
- this formulation decomposes y as $x + v$ by penalizing deviations of both x and v from their known structures
- $\lambda > 0$ weighs the two penalizations

We can use the constraint $y = x + v$ to eliminate the variable v:

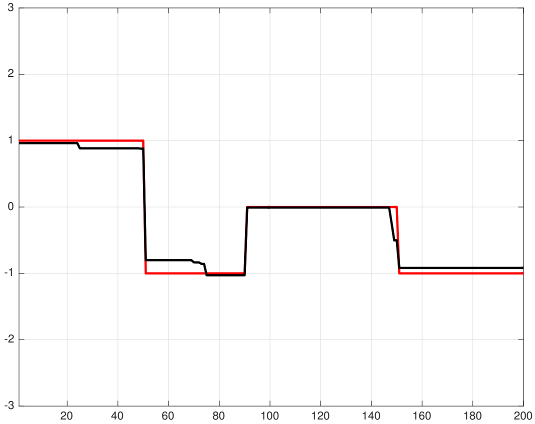
$$\underset{x}{\text{minimize}} \quad \underbrace{\frac{1}{2}\|y - x\|^2 + \lambda \sum_{t=2}^{T} |x(t) - x(t-1)|^p}_{f(x)}$$

We will see the performance of this approach with $p = 2$ and $p = 1$

Solution with $p = 2$ (and $\lambda = 2$):



Solution with $p = 1$ (and $\lambda = 2$):



Denoising a piecewise sawtooth signal

Consider an information signal contaminated with additive noise:

$$\underbrace{\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(T) \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(T) \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} v(1) \\ v(2) \\ \vdots \\ v(T) \end{bmatrix}}_{v}$$

where

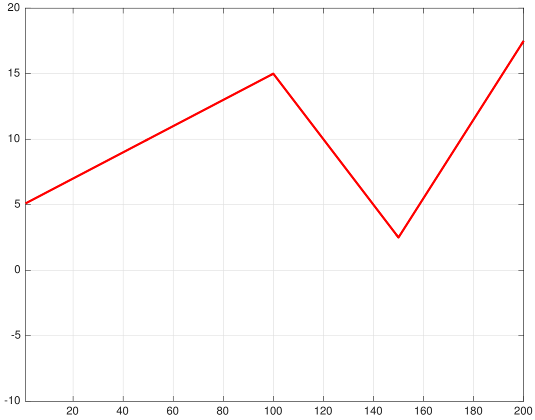
- y is the observed signal (known)
- x is the information signal (unknown)
- v is the additive noise signal (unknown)

Given that you know signal y, how do you guess the unknown signal x?

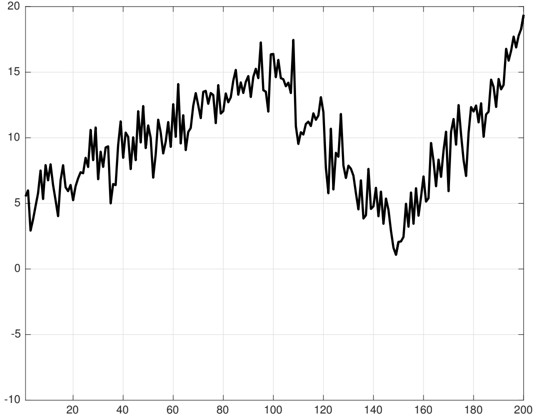
Assume

- the noise signal v is a "small" vector that fluctuates about zero
- the information signal x is piecewise sawtooth (but with unknown switching times and slopes)

Example of an information signal x:

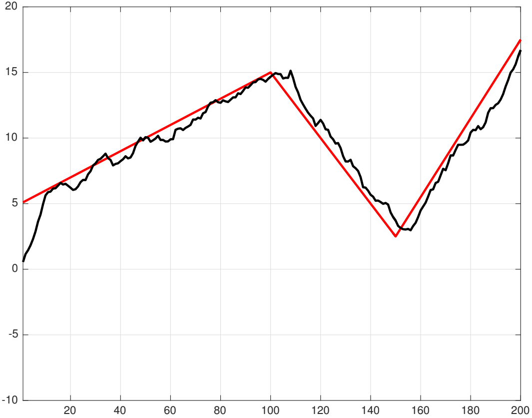


Corresponding example of an observed signal y :



Low-pass filtering approach:

$$\widehat{x}(t) = \frac{1}{10}\left(y(t) + y(t-1) + \cdots + y(t-9)\right)$$



Key-point: x being piecewise constant means

$$(x(t) - x(t-1)) - (x(t-1) - x(t-2)) = 0$$

for most of the ts

Optimization-based approach:

$$\begin{array}{ll} \underset{x,v}{\text{minimize}} & \underbrace{\underbrace{\frac{1}{2}\|v\|^2}_{\text{make } v \simeq 0} + \lambda \underbrace{\sum_{t=2}^{T} |x(t) - 2x(t-1) + x(t-2)|^p}_{\text{make } x(t)-2x(t-1)+x(t-2)=0 \text{ most of the } t\text{s}}}_{f(x,v)} \\ \text{subject to} & y = x + v \end{array}$$

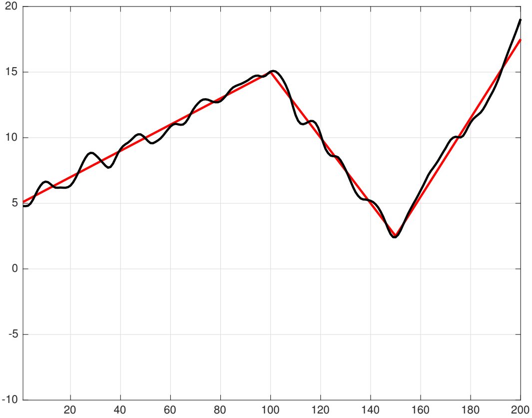
- the optimization variables are x and v
- this formulation decomposes y as $x + v$ by penalizing deviations of both x and v from their known structures
- $\lambda > 0$ weighs the two penalizations

We can use the constraint $y = x + v$ to eliminate the variable v:

$$\underset{x}{\text{minimize}} \quad \underbrace{\frac{1}{2}\left\|y - x\right\|^2 + \lambda \sum_{t=2}^{T} |x(t) - 2x(t-1) + x(t-2)|^p}_{f(x)}$$

We will see the performance of this approach with $p = 2$ and $p = 1$

Solution with $p = 2$ (and $\lambda = 10$):



Solution with $p = 1$ (and $\lambda = 10$):