Summary of lecture 10 (I/II)

- **Model with known input** $u(t)$ a known deterministic input (e.g. control signal),

$$x(t+1) = Ax(t) + Bu(t) + w(t)$$
$$y(t) = Cx(t) + Du(t) + v(t)$$

the only necessary modifications to Kalman filter are

$$\hat{x}(t+1|t) = A\hat{x}(t|t) + Bu(t)$$
$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)\left(y(t) - C\hat{x}(t|t-1) - Du(t)\right)$$

- **Extended Kalman filter** for non-linear state space models.
- **Observability:** Important to check if the state $x(t)$ can be estimated.
- **Implementation**: Assure that P is symmetric and $P > 0$.
- **Design**: Compromize of noise suppression (Q/R small) and tracking speed (Q/R large). Usually, R given by sensor performance, and Q design parameter. Choose P_0 if there is no prior information.

Summary of lecture 10 (II/II)

- **Stationary Kalman filters:** Computationally cheaper. Computed by solving the stationary Riccati equation,

$$\bar{P}_p = A\bar{P}_p A^T - A\bar{P}_p C^T \left(C\bar{P}_p C^T + R\right)^{-1} C\bar{P}_p A^T + Q$$

$$\bar{K} = \bar{P}_p C^T \left(C\bar{P}_p C^T + R\right)^{-1}, \qquad \bar{P}_f = \bar{P}_p - \bar{K}C\bar{P}_p$$

$$\hat{x}(t+1|t) = (A - A\bar{K}C)\hat{x}(t|t-1) + A\bar{K}y(t)$$

$$\hat{x}(t|t) = (A - \bar{K}CA)\hat{x}(t-1|t-1) + \bar{K}y(t)$$

- **Frequency properties:** The stationary Kalman filter has transfer function $H(e^{i\omega T}) = (e^{i\omega T} I - A + \bar{K}CA)^{-1}\bar{K}$ from $y(t)$ to $\hat{x}(t|t)$
- **Using the KF in practice:**
  1. Model the dynamics — $x(t+1) = Ax(t) + Bu(t) + w(t)$
  2. Model the sensors — $y(t) = Cx(t) + Du(t) + v(t)$
  3. Implement a Kalman filter based on the models.
  4. Evaluate performance.

Outline Lecture 11

Adaptive Signal Processing

1. Problem formulation
2. Algorithm format
3. Recursive Least Squares (RLS)
4. Least Mean Squares (LMS)
5. Adaptive estimation using the Kalman filter
6. Simple examples

Review of linear models and least squares methods

- **Linear signal model:** $y(t) = \varphi^T(t)\theta + e(t)$, with parameter θ.
- **Signal estimate:** $\hat{y}(t) = \varphi^T(t)\hat{\theta}$, with parameter estimate $\hat{\theta}$.
- **Least squares method:** Use N measurements $y(t)$ and find the parameter θ by minimizing

$$V_N(\theta) = \frac{1}{N}\sum_{t=1}^{N}\left(y(t) - \hat{y}(t,\theta)\right)^2.$$

  Used to estimate parameters of, e.g., $AR(n)$ models.
- **Stochastic least squares:** Find the parameter θ by minimizing

$$V_N(\theta) = \frac{1}{2}E\left[\left(y(t) - \hat{y}(t,\theta)\right)^2\right],$$

  i.e. minimize the estimation error variance. Used to derive, e.g., the Wiener filter.

Recursive Least Squares (RLS)

- The following cost function is minimized

$$V_t(\theta) = \sum_{k=1}^{t} \lambda^{t-k} \left(y(k) - \varphi^T(k)\theta \right)^2 .$$

  Compare to Lec 6, slide 5. Forgetting factor is only difference, i.e., RLS is Least Squares (cf. Lec 6) with a forgetting factor.

- The recursive solution is given by

$$\begin{aligned}
\hat{\theta}(t) &= \hat{\theta}(t-1) + K(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1)) \\
K(t) &= P(t)\varphi(t) \\
P(t) &= \frac{1}{\lambda}\left(P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)} \right)
\end{aligned}$$

  where $0 < \lambda < 1$ is the forgetting factor.

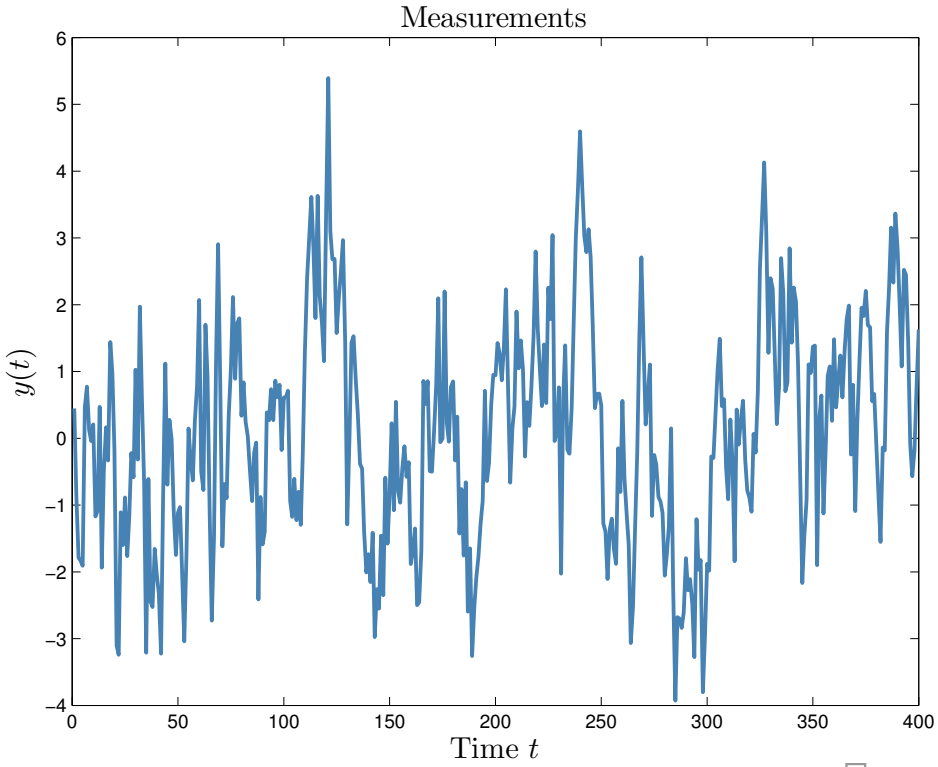
- **Forgetting factor:** Lower/higher $\lambda \Rightarrow$ faster/slower adaptation and higher/lower noise sensitivity.

Linear model example

- Consider a simple $AR(1)$ model,

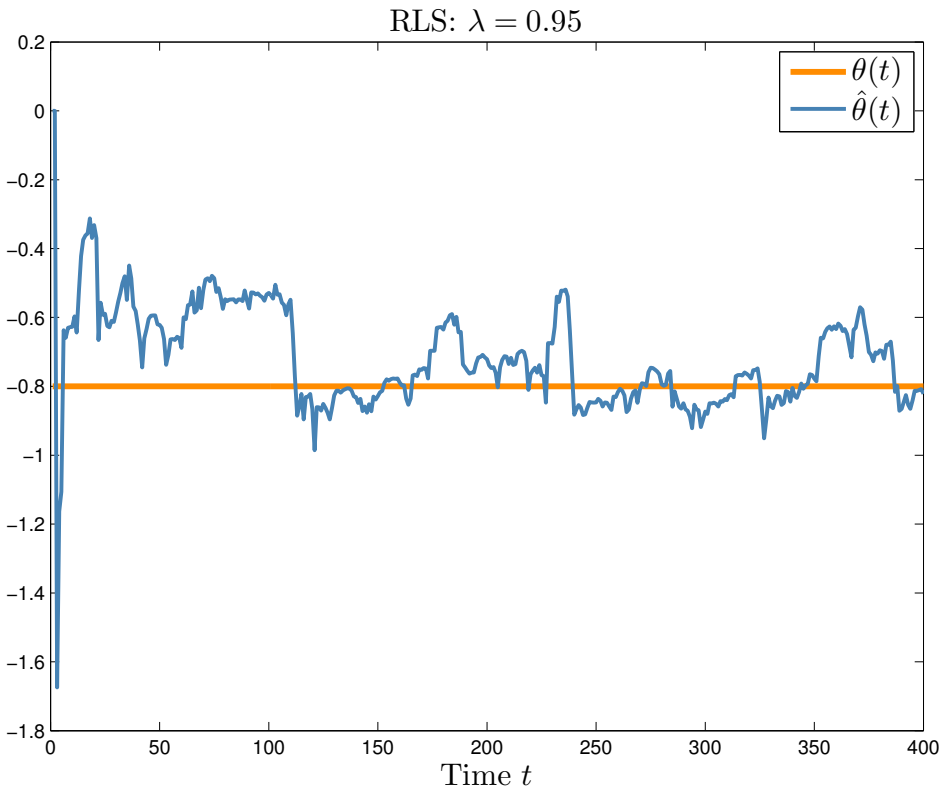
$$y(t) = \varphi^T(t)\theta(t) + e(t)$$
$$\varphi(t) = -y(t-1)$$
$$\theta(t) = a_1 = -0.5$$

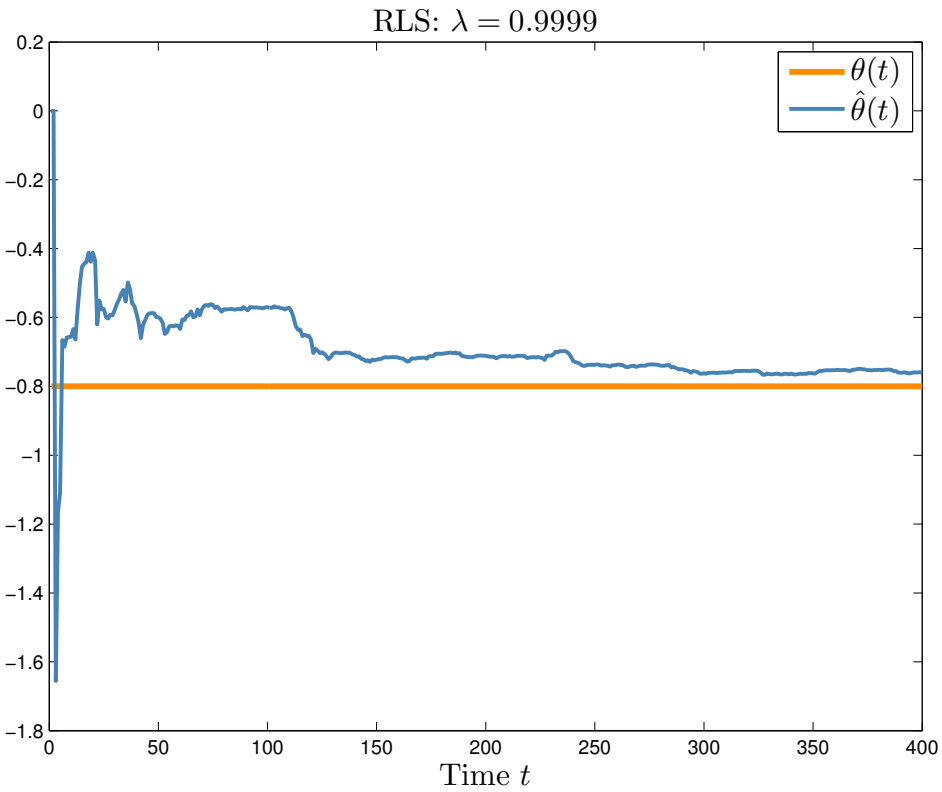
- We have a realization with N measurements



RLS: example

- The RLS algorithm applied to the $AR(1)$ data:





Least Mean Square (LMS)

- The following cost function is minimized

$$V_t(\theta) = \frac{1}{2} E\left[\left(y(t) - \varphi^T(t)\theta\right)^2\right].$$

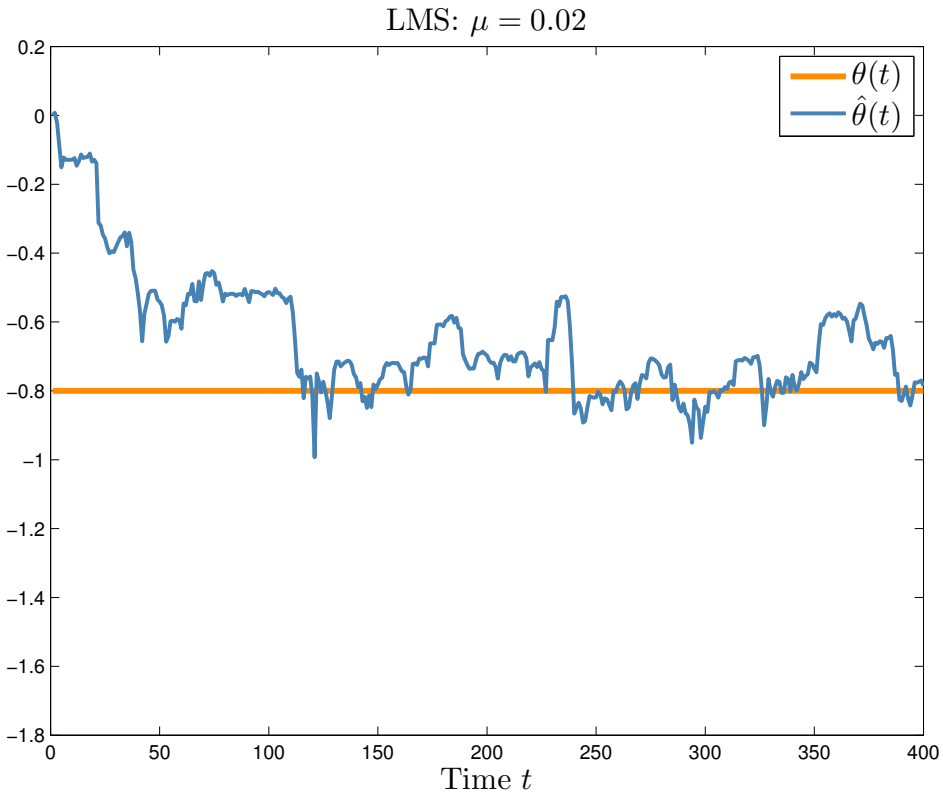
- The recursive solution is given by

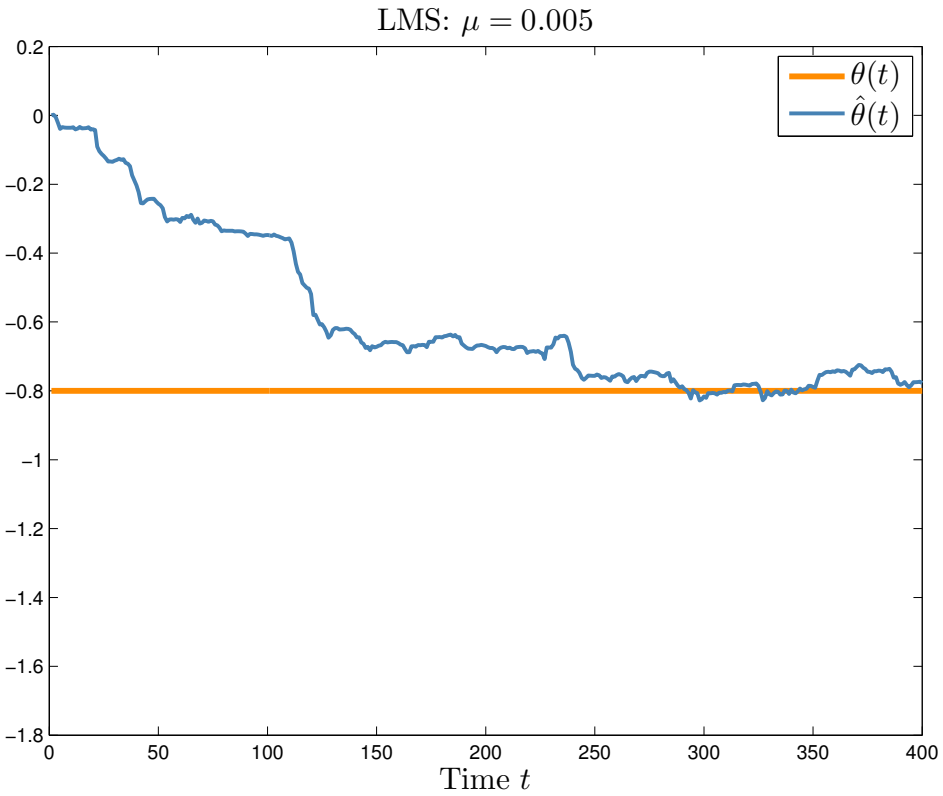
$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$
$$K(t) = \mu\varphi(t)$$

- **Step length:** $\mu > 0$. Higher/lower $\mu \Rightarrow$ faster/slower adaptation and higher/lower noise sensitivity.

LMS: example

- The LMS algorithm applied to the $AR(1)$ data:





Kalman filter adaptation

- Time variable state space model (random walk dynamics),

$$\theta(t+1) = \theta(t) + w(t) \qquad Q(t) = \mathrm{Cov}(w(t))$$
$$y(t) = \varphi^T(t)\theta(t) + e(t) \qquad R(t) = \mathrm{Cov}(e(t))$$

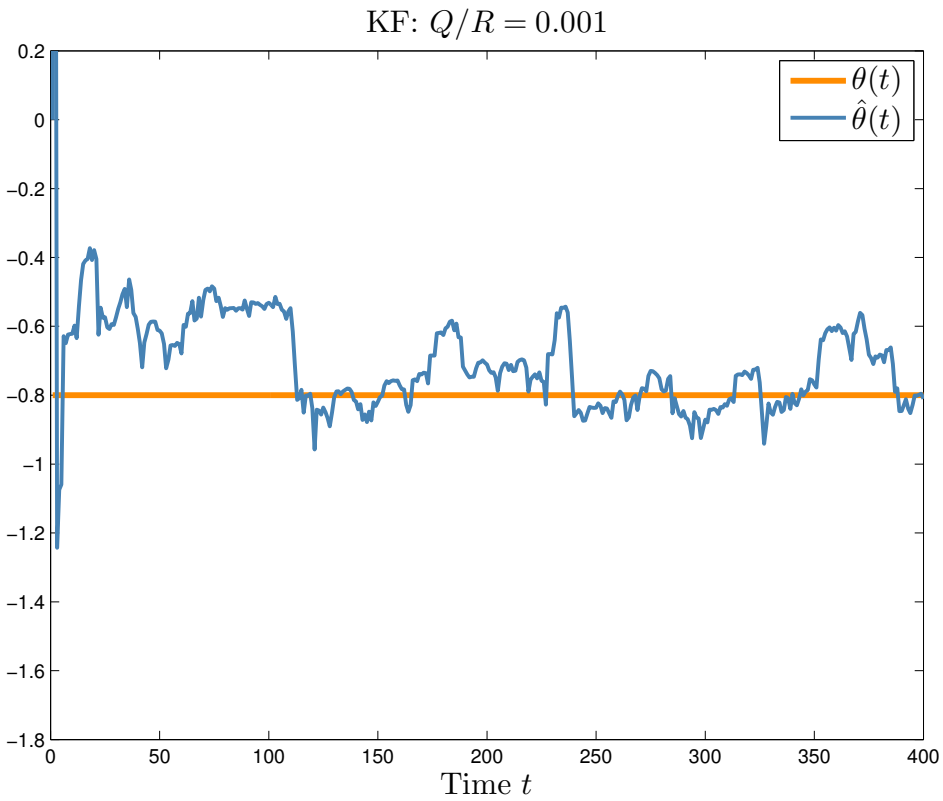
- The recursive solution is given by

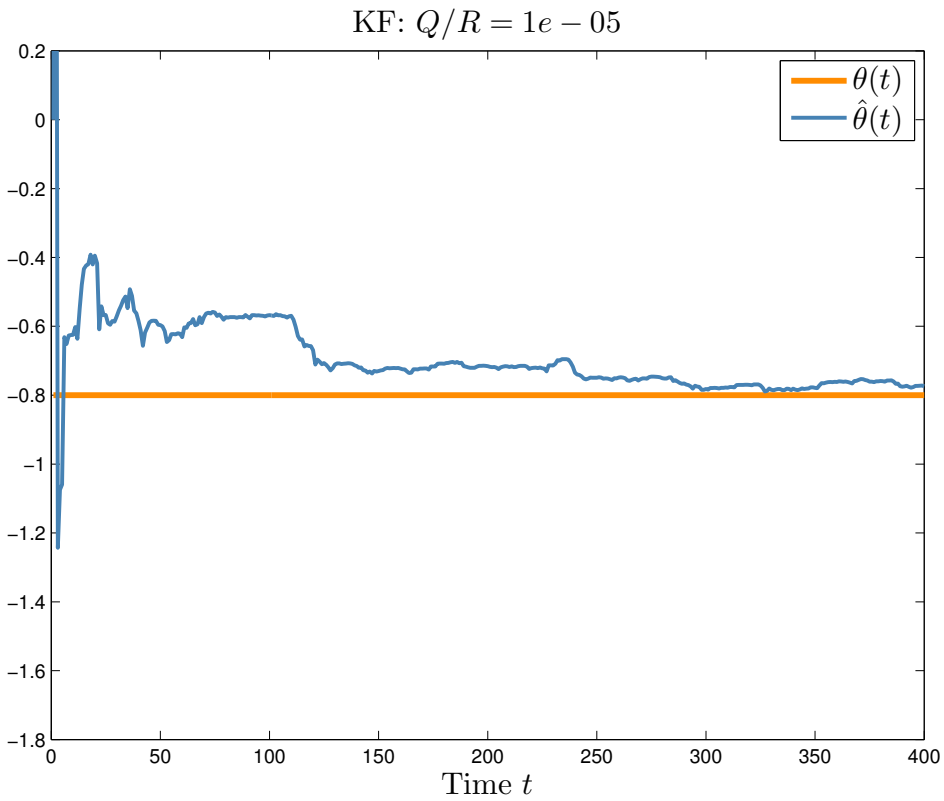
$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$
$$K(t) = P(t-1)\varphi(t)\left(\varphi^T(t)P(t-1)\varphi(t) + R(t)\right)^{-1}$$
$$P(t) = P(t-1) - K(t)\varphi^T(t)P(t-1) + Q(t)$$

- Noise covariances: Q and R. Let R be fixed, then larger/smaller Q $\Rightarrow$ faster/slower adaptation and higher/lower noise sensitivity.

Kalman filter: example

- The Kalman filter applied to the $AR(1)$ data:





Summary of Lecture 11

- **Recursive Least Squares:** The RLS algorithm is based on minimizing the sum of squared errors.
- **Forgetting factor:** Used in RLS to "remove" influence of old measurements. Used to tune the algorithm.
- **Least Mean Squares:** The LMS algorithm is a stochastic gradient descent method. Minimizes the error variance.
- **Step length:** Used in LMS, determines how large the gradient step is in each iteration. Used to tune the algorithm.
- **Normalized LMS:** LMS algorithm with a normalized step length.
- **Adaptation using KF:** Model parameter changes as a random walk, and use a Kalman filter.
- **Adaptation vs noise:** Ideally we want our recursive algorithm to have fast adaptation, and low noise sensitivity. Practically, these are conflicting objectives and we have to compromise between the two.