System Identification

6.435

SET 11

- Computation
- Levinson Algorithm
- Recursive Estimation

Munther A. Dahleh

Computation

<u>Least Squares</u>: QR factorization

$$V_N = |Y - \Phi\theta|^2$$

$$\Phi = T^T \begin{bmatrix} Q \\ 0 \end{bmatrix} \qquad T^T T = I$$

$$V_N = \left| T^T \left(TY - \begin{pmatrix} Q \\ 0 \end{pmatrix} \theta \right) \right|_2^2 = \left| TY - \begin{pmatrix} Q \\ 0 \end{pmatrix} \theta \right|_2^2$$

$$= \left| \begin{pmatrix} L \\ M \end{pmatrix} - \begin{pmatrix} Q \\ 0 \end{pmatrix} \theta \right|_2^2$$

$$= |L - Q\theta|^2 + |M|^2$$

$\boldsymbol{Q}$ is invertible and

$\hat{\theta}_N = Q^{-1} L$

error $|M|_2^2$

Initial Conditions:

$$\Phi(t) = \begin{bmatrix} z(t-1) \\ \vdots \\ z(t-n) \end{bmatrix} \qquad z = \begin{bmatrix} -y(t) \\ u(t) \end{bmatrix} \text{ or } \begin{bmatrix} -y(t) \end{bmatrix}$$

$$R(N) = \frac{1}{N} \sum_{t=1}^{N} \Phi(t)\Phi^T(t)$$

$$= \frac{1}{N} \sum_{t=1}^{N} \begin{pmatrix} z(t-1) \\ \vdots \\ z(t-n) \end{pmatrix} \begin{pmatrix} z^T(t-1) & \dots & z^T(t-n) \end{pmatrix}$$

$$R_{ij}(N) = \frac{1}{N} \sum_{t=1}^{N} z(t-i)z^T(t-j)$$

What about initial conditions?

Solution 1: sum starts $t = n + 1 \to N$

appropriately shifted, assume data is available at $t > -n$

(Covariance method)

Solution 2:

assume $z(-n+1), \ldots, z(0) = 0$

and $z(N+1), \ldots, z(N+n) = 0$

augment the sum to $N + n$

(autocorrelation method).

In the 2nd case

$$R_{ij}(N) = \frac{1}{N} \sum_{t=1}^{N+n} z(t-i) z^T(t-j)$$

$$= \frac{1}{N} \sum_{s=1-j}^{N-j+n} z(s-(i-j)) z^T(s)$$

$$= \frac{1}{N} \sum_{s=1}^{N+n} z(s-(i-j)) z^T(s)$$ only depends on $i-j$.

$$R_\tau(N) = \frac{1}{N}\sum_{t=1}^{N+n} z(t-\tau)z^T(t)$$

$$= \frac{1}{N}\sum_{t=\tau}^{N} z(t-\tau)z^T(t)$$ Block Toeplitz.

<u>Structure allows for fast computations</u>

<u>AR model</u> of order n

$$y^{(n)}(t-1) = -a_1^n y(t-1) + \ldots - a_n^n y(t-n)$$

$$z(t) = -y(t)$$

Levinson Algorithm

$$R_\tau(N) = \frac{1}{N}\sum_{t=\tau}^{N} y(t-\tau)y(t) = \hat{R}_y(\tau)$$

$$\begin{bmatrix} R_o & \dots & \dots & R_{n-1} \\ & R_o & & \\ & & \ddots & \\ R_{n-1} & \dots & \dots & R_o \end{bmatrix} \begin{bmatrix} a_1^n \\ \vdots \\ \\ a_n^n \end{bmatrix} = \begin{bmatrix} -R_1 \\ \vdots \\ \\ -R_n \end{bmatrix}$$

$$\Updownarrow$$

$$\begin{bmatrix} R_o & R_1 & \dots & R_n \\ R_1 & R_o & & R_{n-1} \\ \vdots & & \ddots & \\ R_n & R_{n-1} & \dots & R_o \end{bmatrix} \begin{bmatrix} 1 \\ a_1^n \\ \vdots \\ a_n^n \end{bmatrix} = \begin{bmatrix} V_n \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

definition

$$\Updownarrow$$

$$
\begin{bmatrix} R_o & R_1 & \dots & R_{n+1} \\ \vdots & & & \\ & & & \\ \vdots & & & \\ R_{n+1} & \dots & & R_o \end{bmatrix}
\begin{bmatrix} 1 \\ a_1^n \\ \vdots \\ a_n^n \\ 0 \end{bmatrix}
=
\begin{bmatrix} V_n \\ 0 \\ \vdots \\ 0 \\ \alpha_n \end{bmatrix} \longrightarrow \text{def. of } \alpha_n
$$

flip flip

$$\Updownarrow$$

$$
\begin{bmatrix} R_o & R_1 & \dots & R_{n+1} \\ \vdots & & & \\ \vdots & & & \\ R_{n+1} & R_n & & R_o \end{bmatrix}
\begin{bmatrix} 0 \\ a_n^n \\ \vdots \\ 1 \end{bmatrix}
=
\begin{bmatrix} \alpha_n \\ \vdots \\ \\ V_n \end{bmatrix}
$$

add $\underbrace{\left(-\frac{\alpha_n}{V_n}\right)}_{\rho_n}$ 1st + 2nd $\Rightarrow$

$$
\begin{bmatrix} \\ \text{same} \\ \\ \end{bmatrix}
\begin{bmatrix} \rho_n \\ a_n^n + \rho_n a_1^n \\ \vdots \\ a_1^n + \rho_n a_n^n \\ 1 \end{bmatrix}
=
\begin{bmatrix} 0 \\ \vdots \\ \\ \\ V_n + \rho_n \alpha_n \end{bmatrix}
$$

<u>Flip</u>:

$$\begin{bmatrix} & & \\ & same & \\ & & \end{bmatrix} \begin{bmatrix} 1 \\ a_1^n + \rho_n a_n^n \\ \vdots \\ a_n^n + \rho_n a_1^n \\ \rho_n \end{bmatrix} = \begin{bmatrix} V_n + \rho_n \alpha_n \\ 0 \\ \vdots \\ \\ 0 \end{bmatrix}$$

Clearly $$a_n^{n+1} = \rho_n$$

$$a_k^{n+1} = a_k^n + \rho_n a_{n-k+1}^n$$

$$V_{n+1} = V_n + \rho_n \alpha_n$$

$$\rho_n = -\frac{\alpha_n}{V_n}$$

$$\alpha_n = R_{n+1} + \sum_{k=1}^{n} a_n^n R_{(n+1-k)}$$

Initial conditions

$$V_1 = R_o - \frac{{R_1}^2}{R_o}$$

$$a_1{}' = -\frac{R_1}{R_o}$$

good reduction. $(4n+1)$ comp $2n^2$

Numerical Methods

- $\min\ V_N\left(\theta, Z^N\right)$

  $\text{sol}\left[f_N\left(\theta, Z^N\right) = 0\right]$

  Both have no analytical solutions in general

- General Procedure

$$\widehat{\theta}(i+1) = \widehat{\theta}(i) + \alpha f^{(i)}$$

step size

direction of the search depends on $V_N\left(\theta(i), Z^N\right)$

- Different Methods
  - f depends on V_N
  - f depends on $V_N, V_N{}'$
  - f depends on $V_N, V_N{}', V_N{}''$ (Newton's)
- Newton's $f^{(i)} = -\left[V_N''\left(\widehat{\theta}(i)\right)\right]^{-1} V_N{}'\left(\widehat{\theta}(i)\right)$
- Quasi – Newton: Approximate $V_N^i{}''$ by $V_N^i{}' - V_N^{i-1}{}'$

Special Schemes: Nonlinear Least Squares

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

  $$V_N{}'\left(\theta, Z^N\right) = \frac{1}{N}\sum_{t=1}^{N}\Psi(t,\theta)\varepsilon(t,\theta)$$

- A family of Algorithms

  $$\widehat{\theta}_N(i+1) = \widehat{\theta}_N(i) - \mu_N(i)\left[R_N(i)\right]^{-1} V_N{}'\left(\widehat{\theta}_N(i), Z^N\right)$$

  - $\mu_N(i)$ step size, chosen so that

    $$V_N\left(\widehat{\theta}_N(i+1), Z^N\right) < V_N\left(\widehat{\theta}_N(i), Z^N\right)$$

– $R_N(i) = \begin{cases} I & \Rightarrow \quad \text{gradient steepest descent} \\ V_N{}''\left(\hat{\theta}_N(i), Z^N\right) & \Rightarrow \quad \text{Newton's} \end{cases}$

$$V_N''\left(\hat{\theta}_N(i), Z^N\right) = \frac{1}{N}\sum_{t=1}^{N} \Psi\left(t, \hat{\theta}(i)\right) \Psi^T\left(t, \hat{\theta}(i)\right) - \underbrace{\frac{1}{N}\sum_{t=1}^{N} \Psi^T\left(t, \hat{\theta}(i)\right) \varepsilon\left(t, \hat{\theta}(i)\right)}_{\text{negligible around min.}}$$

$$V_N''\left(\hat{\theta}_N(i), Z^N\right) \simeq \frac{1}{N}\sum_{t=1}^{N} \Psi\left(t, \hat{\theta}(i)\right) \Psi^T\left(t, \hat{\theta}(i)\right)$$

$\Rightarrow$ Newton-Gauss, Newton-Raphson

For instrumental method

$$\theta_N^{i+1} = \theta_N^i - \mu_N^{(i)} f_N\left(\theta^{i-1}, Z^N\right)$$

Newton-Raphson

$$\theta_N^{i+1} = \theta_N^i - \mu_N \left[{f_N}' \left(\theta^{i-1}, Z^N\right)\right]^{-1} f_N\left(\theta^{i-1}, Z^N\right)$$

Computing the gradient:

ARMAX:

$$C\hat{y} = Bu + (C - A)y$$

diff. with respect to a_k

$$C\frac{\partial}{\partial a_k}\hat{y} = -q^{-k}y(t)$$

diff. w. r. to b_k

$$C\frac{\partial}{\partial b_k}\widehat{y} = -q^{-k}u(t)$$

diff. w. r. to c_k

$$q^{-k}\widehat{y} - C\frac{\partial}{\partial c_k}\widehat{y} = q^{-k}y(t)$$

<u>Recall</u>

$$\Phi^T(t,\theta) = (-y(t-1)\ldots, u(t-1)\ldots, \varepsilon(t-1)\ldots)$$

Re-arrange

$$C(q)\Psi(t,\theta) = \Phi(t,\theta)$$

$$\Rightarrow \Psi(t,\theta) = \frac{1}{C(q)}\Phi(t,\theta)$$

dep. on θ, however assumed stable

Of course a special case for ARX: $\Psi(t,\theta) = \Phi(t,\theta)$

Exercise

Jenkins Black-Box model

State-space model

Recursive Methods

- Computational advantages
- Carry the covariance matrix $\left(\widehat{\theta}_N - \theta_o\right)$ in the estimate.

General form:

$$\widehat{\theta}_t = h(x(t))$$

$$x(t) = H(t, x(t-1), y(t), u(t))$$

Specific form:

$$\widehat{\theta}_t = \widehat{\theta}_{t-1} + \gamma_t Q_\theta(x(t), y(t), u(t))$$

$$x(t) = x(t-1) + \mu_t Q_x(x(t-1), y(t), u(t))$$

Least-square estimate as a recursive estimate

Off line

$$\hat{\theta}_t = \underset{\theta}{\text{argmin}} \sum_{k=1}^{t} \beta(t,k) \left[y(k) - \Phi^T(k)\theta \right]^2$$

$$\hat{\theta}_t = \bar{R}^{-1} f(t)$$

$$\bar{R}(t) = \sum_{k=1}^{t} \beta(t,k)\Phi(k)\Phi^T(k)$$

$$f(t) = \sum_{k=1}^{t} \beta(t,k)\Phi(k)y(k)$$

Recursive Identification (LS)

- Assume

$$\beta(t,k) = \lambda(t)\beta(t-1,k) \qquad \forall \quad 1 \le k \le t-1$$

$$\beta(t,t) = 1$$

Example exponential weight $\beta(t,k) = e^{-a(t-k)}$

means in general

$$\beta(t,k) = \prod_{k}^{t} A(t)$$

$$\bar{R}(t) = \sum_{k=1}^{t} \beta(t,k)\Phi(k)\Phi^T(k)$$

$$= \lambda(t) \sum_{k=1}^{t} \beta(t-1,k)\Phi(k)\Phi^T(k) + \phi(t)\phi^T(t)$$

$$= \lambda(t)\bar{R}(t-1) + \phi(t)\phi^T(t)$$

$$f(t) = \lambda(t)f(t-1) + \phi(t)y(t)$$

$$\hat{\theta}_t = \bar{R}^{-1}f = \bar{R}^{-1}(\lambda(t)f(t-1) + \phi(t)y(t))$$

$$= \bar{R}^{-1}\left(\lambda(t)\bar{R}(t-1)\hat{\theta}_{t-1} + \phi(t)y(t)\right)$$

$$= \bar{R}^{-1}\left(\left(\bar{R}(t) - \phi(t)\phi^T(t)\right)\hat{\theta}_{t-1} + \phi(t)y(t)\right)$$

$$\hat{\theta}_t = \hat{\theta}_{t-1} + \bar{R}^{-1}\phi(t)\left[y(t) - \phi^T\hat{\theta}_{t-1}\right]$$

$$\bar{R}(t) = \lambda(t)\bar{R}(t-1) + \phi(t)\phi^T(t)$$

$\Rightarrow$ A recursive algorithm

Normalized Gain estimate:

$$R(t) = \frac{1}{\gamma(t)}\bar{R}(T) \qquad \gamma(t) = \left(\sum_{k=1}^{t} \beta(t,u)\right)^{-1}$$

$$\frac{1}{\gamma(t)} = \frac{\lambda(t)}{\gamma(t-1)} + 1$$

$$R(t) = \gamma(t)\left[\lambda(t-1)\frac{R(t-1)}{\gamma(t-1)} + \phi\phi^T\right]$$

$$= R(t-1) + \gamma(t)\left[\phi(t)\phi^T(t) - R(t-1)\right]$$

Recursive Algorithm:

$$\varepsilon(t) = y(t) - \varphi^T(t)\widehat{\theta}(t-1)$$

$$\widehat{\theta}(t) = \widehat{\theta}(t-1) + \gamma(t)R^{-1}\phi(t)\varepsilon(t)$$

$$R(t) = R(t-1) + \gamma(t)\left[\phi(t)\phi^T(t) - R(t-1)\right]$$

Properties of Recursive Algorithms

- Need to store $\widehat{\theta}(t-1)$ and $\bar{R}(t-1)$ to compute the next estimate $\widehat{\theta}(t)$.

- $\bar{R}(t)$ is the covariance (an estimate) of $\widehat{\theta}_N$ and hence gives an estimate of the accuracy of $\widehat{\theta}_N$. [Recall that $\text{Cov}\ \widehat{\theta}_N \simeq \frac{1}{N}\left(\bar{E}\phi\phi^T\right)$]

- $\bar{R}$ is symmetric, so you only need to store the lower part of it. (Save on memory)

Recursive Algorithms with Efficient Matrix Conversion

- Define $P(t) = \bar{R}^{-1}(t)$
- Inversion formula

$$(A+BCD)^{-1} = A^{-1} - A^{-1}B\left(DA^{-1}B + C^{-1}\right)^{-1} DA^{-1}$$

- Consider the matrix

$$\bar{R}^{-1}(t) = \Big(\underbrace{\lambda(t)\bar{R}(t-1)}_{A} + \underbrace{\phi}_{B}\underbrace{\phi^T(t)}_{D}\Big)^{-1}$$

$$= \frac{1}{\lambda(t)}\bar{R}^{-1}(t-1) - \frac{1}{\lambda(t)}\bar{R}^{-1}(t-1)\phi(t)\left[\phi^T(t)\frac{1}{\lambda(t)}\bar{R}^{-1}(t-1)\phi(t) + 1\right]^{-1}\frac{\phi^T(t)}{\lambda(t)}$$

$$P(t) = \frac{1}{\lambda(t)} P(t-1) - \frac{1}{\lambda(t)} P(t-1)\phi(t) \left[\phi^T \frac{1}{\lambda(t)} \bar{R}^{-1}(t-1)\phi(t) + 1 \right]^{-1} \frac{\phi^T(t)}{\lambda(t)}$$

$$P(t) = \frac{1}{\lambda(t)} \left[P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{\lambda(t) + \phi^T(t)P(t-1)\phi(t)} \right]$$

$$P(t)\phi(t) = \frac{P(t-1)\phi(t)P(t-1)}{\lambda(t) + \phi^T(t)\phi(t)} = L(t)$$

Recursive Algorithm:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)\left[y(t) - \phi^T(t)\hat{\theta}(t-1)\right]$$

$$L(t) = \frac{P(t-1)\phi(t)}{\lambda(t) + \phi^T(t)P(t-1)\phi(t)}$$

$$P(t) = \frac{1}{\lambda(t)}\left[P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{\lambda(t) + \phi^T(t)P(t-1)\phi(t)}\right]$$

Advantage: no need to compute R^{-1} at each iteration

$P = R^{-1}$ is iterated directly!

- Kalman filter interpretation
  - $L(t)$ is the gain
  - $P(t)$ is the solution of the Riccati equation

$$\begin{cases} \theta(k+1) = \theta(k) \\ y(t) = \Phi^T(t)\theta(t) + v(t) \end{cases}$$

Recursive Instrumental Variable Method

For a fixed, not model dependent Instrument,

$$\theta_t^{IV} = \bar{R}(t)^{-1} f(t)$$

$$\bar{R}(t) = \sum_{k=1}^{t} \beta(t,k)\xi(k)\Phi^T(k)$$

β is some weight

$$f(t) = \sum_{k=1}^{t} \beta(t,k)\xi(k)y(k)$$

You can show:

$$\widehat{\theta}(t) = \widehat{\theta}(t-1) + L(t)\left[y(t) - \Phi^T(t)\widehat{\theta}(t-1)\right]$$

$$L(t) = \frac{P(t-1)\xi(t)}{\lambda(t) + \Phi^T(t)P(t-1)\xi(t)}$$

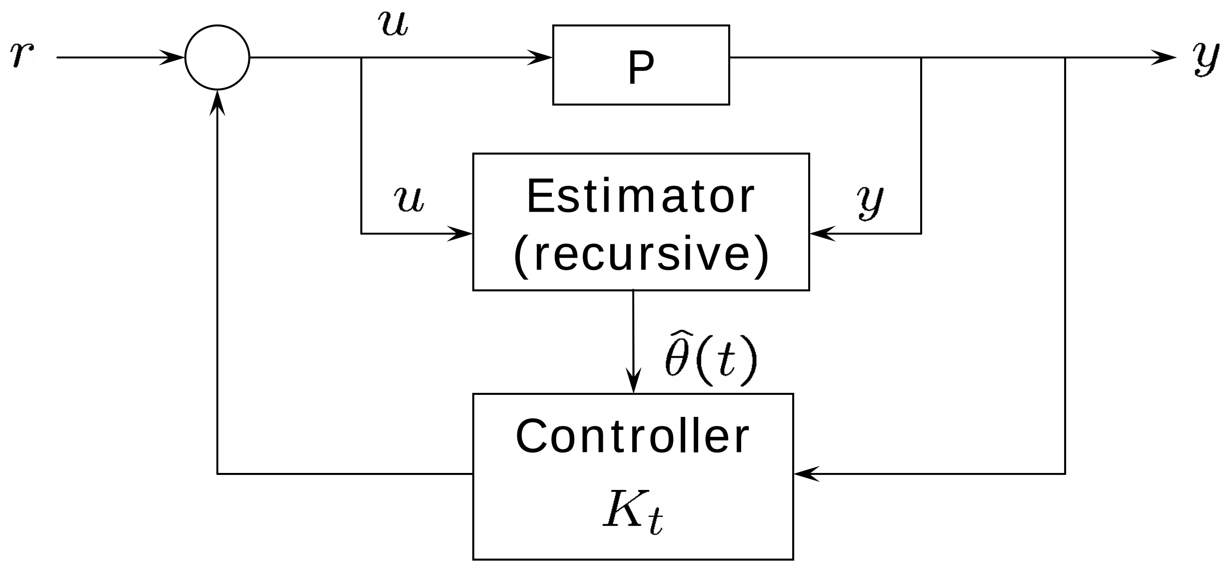
$$P(t) = \frac{1}{\lambda(t)}\left[P(t-1) - \frac{P(t-1)\xi(t)\Phi^T(t)P(t-1)}{\lambda(t) + \Phi^T(t)P(t-1)\xi(t)}\right]$$

where $\lambda(t)$ satisfies (by assumption)

$$\beta(t,k) = \lambda(t)\beta(t-1,k) \quad 1 \leq k \leq t-1$$

$$\beta(t,t) = 1$$

Adaptive Control



- r is a bdd input.
- Estimator: Std least squares
- Controller K_t: Condition $\left(\widehat{P}_t, K_t\right)$ is a stable time-varying system.
- $\Rightarrow$ $y(t)$ is bold for any bdd $r(t)$