Lecture 13: Solution of Lyapunov Equation for Continuous and Discrete Systems

Prof. Krishna R. Pattipati
Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu (860) 486-2890

ECE 6435
Adv Numerical Methods in Sci Comp
Fall 2008
November 12, 2008
Solution of Lyapunov Equation For Continuous and Discrete Systems

- What is a Lyapunov Equation?
- Application of Lyapunov Equation
- Computational methods for solving the Lyapunov Equation
  - Direct method
  - Iterative methods
  - Semi-iterative methods
What is Lyapunov Equation?

- Continuous-time Lyapunov Equation: \( A^T X + XA + S = 0 \)
- Discrete-time Lyapunov Equation: \( X = A^T XA + S \)
- The equations are linear and arise in the context of stability of linear systems.

Lyapunov Equation and Stability of Linear Systems

- Continuous-time system: \( \dot{x} = Ax + Bu; \ y = Cx \)
- Discrete-time system: \( x_{i+1} = Ax_i + Bu_i; \ y_i = Cx_i \)
- Original Lyapunov theorem: "\( \dot{x} = Ax \) is asymptotically stable iff for \( S > 0, \exists \) a PD solution \( X \) for \( A^T X + XA + S = 0 \)."

\[
\Rightarrow X=\int_{0}^{\infty} e^{A^T \sigma} Se^{A \sigma} d\sigma
\]
"For discrete-time systems, $x_{i+1} = Ax_i$ is asymptotically stable iff for $S > 0$, $\exists$ a PD solution $X$ for $X = A^T X A + S$"

$$X = \sum_{i=0}^{\infty} (A^T)^i S A^i$$

In the above cases, $v(x) = x^T X x$ is a Lyapunov function.

Lyapunov equation is used in estimating the rates at which $\|x\| \to 0$

Lyapunov function is used to analyze Lyapunov controllers, observers, etc.

Our interest in Lyapunov equation stems from control and filtering applications rather than stability
Lyapunov Equation & Linear Quadratic Regulator (LQR) Problem

"For the continuous-time system, \( \dot{x} = A x + B u \);

find a linear feedback control \( u(t) = -L x(t) \) that minimizes

\[
J(u) = \int_{0}^{\infty} \left[ x^T(t) Q x(t) + u^T(t) R u(t) \right] dt; Q > 0, R > 0
\]

What does \( J(u) \) mean?

"We want \( x \rightarrow 0 \) from \( x_0 \) without using too much control."

For a given \( L \) computation of \( J(u) \) \( \Rightarrow \) solution of Lyapunov equation

- Suppose we take any feedback control \( u(t) = -L x(t) \), where \( L \) is known
- What is the associated cost?
  - Obviously we need \( \dot{x} = (A - BL)x \) stable, otherwise \( x^T Q x \rightarrow \infty \)
  - Let \( \bar{A}_L = A - BL \Rightarrow x(t) = e^{\bar{A}_L t} x_0 \)
Therefore,
\[
J(u) = \sum_{i=0}^{T} \left[ e^{A_T} Q e^{A_T} + e^{A_T} L^T R L e^{A_T} \right] dt \leq 0
\]
\[
= \sum_{i=0}^{T} e^{A_T} [Q + L^T R L] e^{A_T} dt \leq 0 \quad = x_0 V_L x_0
\]
where \( V_L = "cost\ matrix" \ associated with gain \( L \).

- Note that \( V_L \) satisfies the Lyapunov equation:
  \[
  \dot{V}_L + V_L + Q + L^T R L = 0
  \]

- **How can we pick \( L^* \in V_L^* < V_L \forall L ? \) i.e., \( V_L - V_L^* \geq 0 \)?
  
  => Solving continuous-time Riccati equation!.....Lecture 14

- **Discrete-time case:** 
  "Given a discrete-time system,
  
  \[
  x_{i+1} = A x_i + B u_i \;
  \]

  find a linear feedback control \( u_i = -Lx_i \) that minimizes

  \[
  J(u) = \sum_{i=0}^{\infty} x_i^T Q x_i + u_i^T R u_i "
  \]
• Closed-loop system matrix
  \[ \bar{A}_L = A - BL \]

• State in terms of closed-loop system matrix
  \[ \bar{x}_i = \bar{A}_L \bar{x}_0 \]

• Cost function can be rewritten in terms of closed-loop system matrix as:
  \[
  J(u) = x_0^T \left[ \sum_{i=0}^{\infty} (\bar{A}_L)^i (Q + L^T RL) \bar{A}_L \right] x_0 = x_0^T V_L x_0
  \]

• The cost matrix \( V_L \) satisfies the discrete Lyapunov equation:
  \[
  \bar{A}_L V_L \bar{A}_L + Q + L^T RL = V_L
  \]

• Again the problem of picking \( L^* \) \( \exists V_L^* = \min_L x_0^T V_L x_0 \) involves
  discrete-time Riccati equation.
Lyapunov Equation and the Estimation problem

- Consider a linear continuous-time system:
  \[ \dot{x} = Ax + Ew \]  
  where \( w \) is zero mean Gaussian noise  
  with covariance matrix \( E\{w(t)w(\tau)\} = W\delta(t-\tau) \)

- \( \Rightarrow X(t) = \text{cov}\{x(t)\} = E\{x(t)x^T(t)\} \), where  
  \[ \dot{X}(t) = AX(t) + X(t)A^T + EWE^T \]

- If \( A \) is stable, then in the steady state  
  \[ 0 = AX +XA^T + EWE^T \]

- For discrete-time systems  
  \[ x_{i+1} = Ax_i + Ew_i \]  
  where \( w_i \) is zero mean white Gaussian noise sequence  
  with covariance matrix \( E\{w_i w_i^T\} = W\delta_{ij} \)

- \( \Rightarrow \) s.s covariance of \( x \) denoted by \( X \) satisfies:  
  \[ X = AXA^T + EWE^T \]
Consider a continuous-time linear system:

\[ \dot{x} = Ax + Bu + Ew \]

Measurement equation

\[ y(t) = C x(t) + v(t) \]

Objective: Generate estimate \( \hat{x}(t) \) continuously.

If there is no noise and we knew \( x_0 \), then

\[ \Rightarrow d \hat{x}(t) / dt = A \hat{x}(t) + Bu(t); \quad \hat{x}(0) = x_0 \]

\[ \dot{e}(t) = A e(t) \Rightarrow e(t) = e^{At} e_0 = 0; \quad e(t) = x(t) - \hat{x}(t) \]

But, if \( e_0 \neq 0 \) and \( A \) is unstable, then \( \|e(t)\| \to \infty \).

Solution: Use measurements to stabilize.

\[ \text{e.g., } d \hat{x}(t) / dt = A \hat{x}(t) + Bu(t) + K[y(t) - C \hat{x}(t)] \]

residual \( y(t) = y(t) - C \hat{x}(t) = Ce(t) + v(t) \)
In fact, we can accomplish more than stability!! We can choose $K$ to minimize a performance criterion. "minimize $\text{cov}\{e(t)\}$ or MMSE in s.s".

Now $\dot{e}(t) = (A - KC)e(t) + Ew - Ky(t)$

let $\Sigma = \text{cov}[e(t)]$ in ss $\Rightarrow 0 = (A - KC)\Sigma + \Sigma(A - KC)^T + EWE^T + KVK^T$

We need $(A - KC)$ stable, so pick $K^* \in \Sigma_{K^*} < \Sigma_K \forall K \neq K^*$

- Sensitivity Analysis

$0 = AX_i + X_iA^T + (A_iX + E_iWE^T + EWE_i^T + XA_i^T); \quad X_i = \frac{\partial X}{\partial \theta_i}$

output feedback problem, insensitive control design, PDE,....
Computational Techniques

- Direct methods ⇒ solve $Ax = \underline{b}$
- Iterative methods ⇒ sum up terms. Use doubling schemes
- Semi-iterative methods ⇒ use QR method to reduce $A$ to special form and then use $Ax = \underline{b}$ on the modified matrix.

1) **Direct Method**: Equation has $n(n+1)/2$ unknowns. Organize $X$ and $S$ as vectors.

   - Rewrite as $A_V x_V = -S_V$;

   **Example**:

   \[
   \begin{bmatrix}
   a_{11} & a_{12} \\
   a_{21} & a_{22}
   \end{bmatrix}
   \begin{bmatrix}
   x_{11} \\
   x_{12}
   \end{bmatrix}
   =
   \begin{bmatrix}
   a_{11} x_{11} + a_{21} x_{12} \\
   a_{12} x_{11} + a_{22} x_{12}
   \end{bmatrix}
   \Rightarrow
   \begin{bmatrix}
   2a_{11} & 2a_{21} & 0 \\
   a_{12} & a_{11} + a_{22} & a_{21} \\
   0 & 2a_{12} & 2a_{22}
   \end{bmatrix}
   \begin{bmatrix}
   x_{11} \\
   x_{12} \\
   x_{22}
   \end{bmatrix}
   =
   \begin{bmatrix}
   s_{11} \\
   s_{12} \\
   s_{22}
   \end{bmatrix}
   \]

   - need $\lambda_k (A_V) = \lambda_i + \lambda_j \neq 0$
Can see as follows:
- \( A = U \Lambda U^{-1}; A^T = V \Lambda V^{-1} \); Note: \( V = (U^{-1})^T \) and \( V^{-1} = U^T \)
- \( XU \Lambda U^{-1} + V \Lambda V^{-1} X = -S \)
- \( V^{-1} XU \Lambda + \Lambda V^{-1} XU = -V^{-1} SU \)
- define \( Y = V^{-1} XU = U^T XU \) and \( \tilde{S} = V^{-1} SU = U^T SU \). then
\[ Y\Lambda + \Lambda Y = -\tilde{S} \]
\[ \Rightarrow y_{ij} = -s_{ij} / (\lambda_i + \lambda_j) \]
\[ \Rightarrow \text{need } \lambda_i + \lambda_j \neq 0 \]

Can be solved via **LU decomposition**

Can solve for multiple \( S_i \)

Direct method requires \( O(n^6 / 24) \) operations.

Very **bad approach** for \( n \geq 6 \) due to round-off errors and/or CPU time.

**Accuracy** is not very well controlled.
2) "Iterative Method" ....Generate $X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow \ldots \rightarrow X_n \rightarrow X$.

- First, consider the discrete Lyapunov equation:
  \[ X = A^T X A + Q \]
  - We know that the solution is $X = \sum_{i=0}^{\infty} (A^T)^i QA^i$.
  - So, sum via the doubling algorithm.

Convergence test $\| \Delta X \| \leq TOL \| X \|$ 
$\Delta x_{ii}, i = 1, 2, \ldots$
TOL = $10^{-5}$ or so 
or test on diagonal elements

- Convergence rate $\| \Delta X_k \| \leq \| (A^2)^k \| \cdot \| X_k \|$
- Actually growth is governed by $\lambda_{\text{max}} (F)$
- So, $\| \Delta X_k \| \leq \left| \lambda_{\text{max}} (A) \right|^k \| X \|$
- Rarely, if ever, do we need more than 10 iterations ($k = 10$ will handle $\lambda_{\text{max}} \approx .99$)
  So, expect $< 25n^3$ operations.
Application to Continuous Lyapunov Equation

Recall $X$ is s.s solution of $dX / dt = A^T X + XA + S$

There are basically two approaches

(a) Discrete-time representation

(b) Bilinear transformation

(a) **Discrete-time representation**

$$X(t + \Delta) = e^{A^T \Delta} X(t) e^{A\Delta} + \int_{0}^{\Delta} e^{A^T \sigma} S e^{A \sigma} d\sigma$$ valid for $\forall \Delta$.

- So, pick $\Delta \in \Delta \leq 0.5 \|A\|$ and use algorithms for $e^{A\Delta}$ and $\int_{0}^{\Delta} e^{A^T \sigma} S e^{A \sigma} d\sigma$

- Then use **doubling scheme** with $F = e^{A\Delta}$ and $Q = \int_{0}^{\Delta} e^{A^T \sigma} S e^{A \sigma} d\sigma$

- Need $\sim 25n^3$ MADDs just for set up.

- Also, **truncation errors** in $e^{A\Delta}$ and $Q$ will give same order of magnitude errors in $X$

- Convergence rate depends on $\left| \lambda_{\max}(e^{A\Delta}) \right| = e^{\sigma_{\min}(A)\Delta}$; $\sigma_{\min} = \min$ real part $< 0$
– So, bigger $\Delta \Rightarrow$ faster convergence rate; $\Delta$ too small $\Rightarrow$ trouble.

- **Comments:**
  - Very simple method (needs only matrix multiplication routine)
  - Safe and robust, but too costly in initialization
  - Keep $\Delta \geq 1/\|A\|

- **(b) Bilinear transformation:**
  - In (a), we have used an exponential transformation:
    \[ \Phi = e^{A\Delta} \]
  - **Idea:** Suppose, we define $\Phi = (\tau A + I)(\tau A - I)^{-1}; \tau > 0$
    since functions of $A$ commute, we also have:
    \[ \Phi = (\tau A - I)^{-1}(\tau A + I) \]
    This is called **bilinear transformation**. Why?
  - Because if the transformation is solved for $A$: 

(τA − I)Φ = τA + I
τAΦ − Φ = τA + I \implies τA(Φ − I) = Φ + I
⇒ A = (Φ + I).(Φ − I)^{-1} / τ
⇒ same form as original ⇒ Bilinear

• If we substitute A into \(A^T X + XA + S = 0\), we obtain
\(X(Φ + I)(Φ − I)^{-1} / τ + (Φ^T − I)^{-1}(Φ^T + I)X / τ + S = 0\)
\((Φ^T − I)X(Φ + I) + (Φ^T + I)X(Φ − I) + τ(Φ^T − I)S(Φ + I) = 0\)
⇒ \(2Φ^T XΦ − 2X + τ(Φ^T − I)S(Φ + I) = 0\)
Note that since \(Φ = (τA − I)^{-1}(τA + I)\)
\(= (τA − I)^{-1}(τA − I + 2I)\)
⇒ \(Φ = I + 2(τA − I)^{-1}\)
⇒ \(Φ^T XΦ − X + 2τ(τA^T − I)^{-1}S(τA − I)^{-1} = 0\)
⇒ So, this is a discrete-time Lyapunov equation with
\(Q = 2τ(τA^T − I)^{-1} S(τA − I)^{-1}\)
How to pick \( \tau > 0 \)?

- Recall that \( \lambda_i(\Phi) = (\tau \lambda_i(A) + 1) / (\tau \lambda_i(A) - 1) \).
- We would like to pick \( \tau \) so \( \lambda_{\max}(\Phi) \) is minimized to speed up convergence.
- For real roots \( \frac{1}{\tau^*} = \sqrt{\lambda_{\min}(A) \lambda_{\max}(A)} \) \( \sim \) geometric mean
- For arbitrary case \( 1/\tau \sim |\text{tr}(A)|/n \) can be argued "heuristically"

since want \( \tau \approx \frac{1}{|\lambda_1|} \). Use \( \tau = \min[1, 4/n \sum_i |a_{ii}|] \)
Algorithm using bilinear transformation

1. Pick $\tau$
2. Compute:
   \[ \Phi = I + 2(\tau A - I)^{-1} \]
   \[ Q = 2\tau(\tau A^T - I)^{-1} S(\tau A - I)^{-1} \]
3. Solve for $X$ using doubling scheme

- Note that the set up requires $\approx 1.5$ multiplications +1 inversion $\approx 2.5n^3$ operations.
- Excellent method for solving Lyapunov equation!

Can extend to generalized Lyapunov equation in a straightforward manner

\[ AX + XB + C = 0; A, B \text{ stable} \]

$A$ is $n \times n$, $B$ is $m \times m$, $C$ is $n \times m$ and $X$ is $n \times m$
3) Semi iterative methods

Bartels and Stewart "Solution of matrix equation $AX + XB = C$"

- Consider $A^T X + XA + S = 0$  
  
  Idea: Find an orthogonal matrix $Q \triangleright Q^T AQ = \text{upper Schur form}$

$$Q^T AQ = \tilde{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{22} & \cdots & A_{2p} \\ & & \ddots & \ddots \\ A_{pp} & & & \end{bmatrix}$$

diagonals are 1x1 or 2x2 blocks

and there are $p$ such blocks

- Pre- and post multiply(*) by $Q^T$ and $Q$ to obtain

$$Q^T XQ \cdot Q^T AQ + (Q^T AQ)^T Q^T XQ + Q^T SQ = 0$$

$$\Rightarrow \tilde{X} \tilde{A} + \tilde{A}^T \tilde{X} + \tilde{S} = 0$$

- Q: Is it easier to solve this equation?

- A: Yes!! Can be solved in pieces. Recall forward elimination!!
− Partition blocks of $X$ conformal with $A$ so that

$$
\begin{bmatrix}
X_{11} & X_{12} & \ldots & X_{1p} \\
X_{21} & X_{22} & \ldots & X_{2p} \\
X_{k1} & X_{k2} & \ldots & X_{kp} \\
X_{p1} & \ldots & \ldots & X_{pp}
\end{bmatrix}
= \begin{bmatrix}
A_{11} & A_{12} & A_{1L} & A_{1p} \\
0 & A_{22} & \ldots & A_{2p} \\
0 & \ldots & \ldots & A_{pp}
\end{bmatrix}
+ \begin{bmatrix}
X_{11} & X_{12} & \ldots & X_{1p} \\
X_{21} & X_{22} & \ldots & X_{2p} \\
X_{k1} & X_{k2} & \ldots & X_{kp} \\
X_{p1} & \ldots & \ldots & X_{pp}
\end{bmatrix}
+ \begin{bmatrix}
S_{11} & \ldots & S_{1p} \\
S_{21} & \ldots & S_{2p} \\
S_{p1} & \ldots & S_{pp}
\end{bmatrix}
= 0
$$

− Solve for each sub-block $X_{kl}$.
− Note that:

1. $X_{11}A_{11} + A_{11}^TX_{11} + S_{11} = 0 (k = 1, l = 1)$
   $\Rightarrow X_{11}$ via algebraic symmetric formula. $X_{11}$ either 2x2 or 1x1.
2. $X_{11}A_{12} + X_{12}A_{22} + A_{11}X_{12} + S_{12} = 0 (k = 1, l = 1)$
   $X_{11}A_{12}$ is known, so the unknown $X_{12}$ can be solved via:
Must solve an equation of the form $XA + B^T X + C = 0$

1. $A$ 1x1 and $B$ 1x1
2. $A$ 1x1 and $B$ 2x2
3. $A$ 2x2 and $B$ 1x1
4. $A$ 2x2 and $B$ 2x2

$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$; $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$; $C = \begin{bmatrix} c_1 & c_4 \\ c_2 & c_3 \end{bmatrix}$; $X = \begin{bmatrix} x_1 & x_4 \\ x_2 & x_3 \end{bmatrix}$

- For each case, we can solve for $x_1, x_2, x_3$ algebraically by expanding equation.
- Solution exists provided $\lambda_i(A) + \lambda_j(B) \neq 0$.

Case 1: $x_1 = -c_1 / (a_{11} + b_{11})$

Case 2: $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} a_{11} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = 0$

Case 3: Similar to 2
Case 4:
\[
\begin{bmatrix}
 a_{11} + b_{11} & b_{21} & 0 & a_{21} \\
 b_{12} & a_{11} + b_{22} & a_{21} & 0 \\
 0 & a_{12} & a_{22} + b_{22} & b_{12} \\
 a_{12} & 0 & b_{21} & a_{22} + b_{11}
\end{bmatrix}
\begin{bmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4
\end{bmatrix}
= -
\begin{bmatrix}
 c_1 \\
 c_2 \\
 c_3 \\
 c_4
\end{bmatrix}
\]

\[\Rightarrow \text{So, we work across rows } x_{11} \rightarrow x_{12} \rightarrow \ldots x_{1p}, \quad x_{22} \rightarrow \ldots\]

- In general, consider block \(kl\); \(l \geq k\). solve by writing subequation for \(X_{kl}\).

Have \(X_{ij}\) that have already been computed

\[
\sum_{i=1}^{l} X_{ki} A_{il} + \sum_{j=1}^{k} A_{jk}^{T} X_{jl} + S_{kl} = 0
\]

\[
X_{kl} A_{ll} + A_{kk}^{T} X_{kl} + S_{kl} + \sum_{i=1}^{l-1} X_{ki} A_{il} + \sum_{j=1}^{k-1} A_{jk}^{T} X_{jl} = 0
\]

- Solution for subblock is fast and accurate via \(LU\) decomposition
- Then, desired solution \(X = Q \tilde{X} Q^{T}\).
• Computational load = \(2n^3 + 4\sigma n^3 + \frac{7n^3}{2}\)
  
  - \(A \rightarrow\) Hessenberg form \(\approx 2n^3\)
  - Hessenberg \(\rightarrow\) Schur form \(\approx 4\sigma n^3\)
  - Algebraic solution + forming \(S, X\) from \(\tilde{X} \approx \frac{7n^3}{2}\)

• Note if want to solve \(XA + A^TX + S_i = 0, i = 1, 2, \ldots\) This can be accomplished in \(\approx \frac{7n^3}{2}\) (~30% of time to solve for \(i = 1\)).

• Solution of adjoint equation \(XA^T + AX + C = 0\) after having solved original
  
  - This arises in optimal output feedback and insensitive control system design problems
  - Have \(A = Q^TAQ =\) upper Schur form
  - So, need to solve

\[
Q^TXQ\left(Q^TAQ\right)^T + Q^TAQQ^TXQ + Q^TCQ = 0
\]

\[
\tilde{X} = Q^TXQ; \quad \tilde{A}^T = Q^TAQ =\) lower Schur form
\]

- Can transform \(\tilde{A}^T\) to upper Schur form via \(E\tilde{A}^TE\), where

\[
\begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\]

\(n \times n\) Exchange matrix. \(E\) is orthogonal and symmetric

\[
E^2 = I\quad \text{and} \quad E^{-1} = I
\]
• So, solve

\[ E\tilde{X}E\tilde{A}^T E + E\tilde{A}EE\tilde{X}E + E\tilde{C}E = 0 \]

\[ \Rightarrow \tilde{X}_1A_1 + \tilde{A}_1^T \tilde{X}_1 + \tilde{C}_1 = 0 \]

☐ Algorithm steps:

1) \[ \tilde{A}_1 = E\tilde{A}^T E = \tilde{A}^T \] with rows and columns in reverse order.

2) \[ E\tilde{C}E \]

3) Solve for \( \tilde{X}_1 \)

4) \[ X = QE\tilde{X}_1EQ^T \]

☐ Advantages of Barter-Stewart algorithm:

1) Faster than iterative method

2) Excellent for repeated solutions and adjoint (also \( C \) need not be equal to \( C^T \))

3) Can solve when \( A \) is not stable. Need only \( \lambda_i + \lambda_j \neq 0 \). Of course, solution won’t be PD in this case.
Problems with Bartels-Stewart

Problems:

1) Disappointing accuracy \(\approx 4\) digits vs 5 digits for iterative. Why? because scheme generated to accuracy of QR and orthogonality of \(Q\).

What to do? Use iterative improvement.

Let \(X_1=\)solution via Bartels-Stewart’s algorithm.

Let the true solution be \(X = X_1 + \delta X\)

\((X_1 + \delta X)A + A^T(X_1 + \delta X) + C = 0\)

\[\Rightarrow \delta XA + A^T\delta X + (C + X_1A + A^TX_1) = 0\]

\(C + X_1A + A^TX_1 \rightarrow\) residual must be computed in DP

solve for \(\delta X\) using Bartels-Stewart’s algorithm.

already have \(A \Rightarrow 7/2n^3\) ops + 1 matrix multiplication

2) In case when \(A =\) stable and \(C \geq 0\), \(X\) need not be PD.

3) More storage needed (\(\approx 2-3\) \(n^2\) locations)

4) More software needed (recall the need for QR algorithm to compute upper Schur form)
Summary

- Background on Lyapunov Equation
- Application of Lyapunov Equation
- Computational methods for solving the Lyapunov Equation
  - Direct method
  - Iterative methods
  - Semi-iterative methods